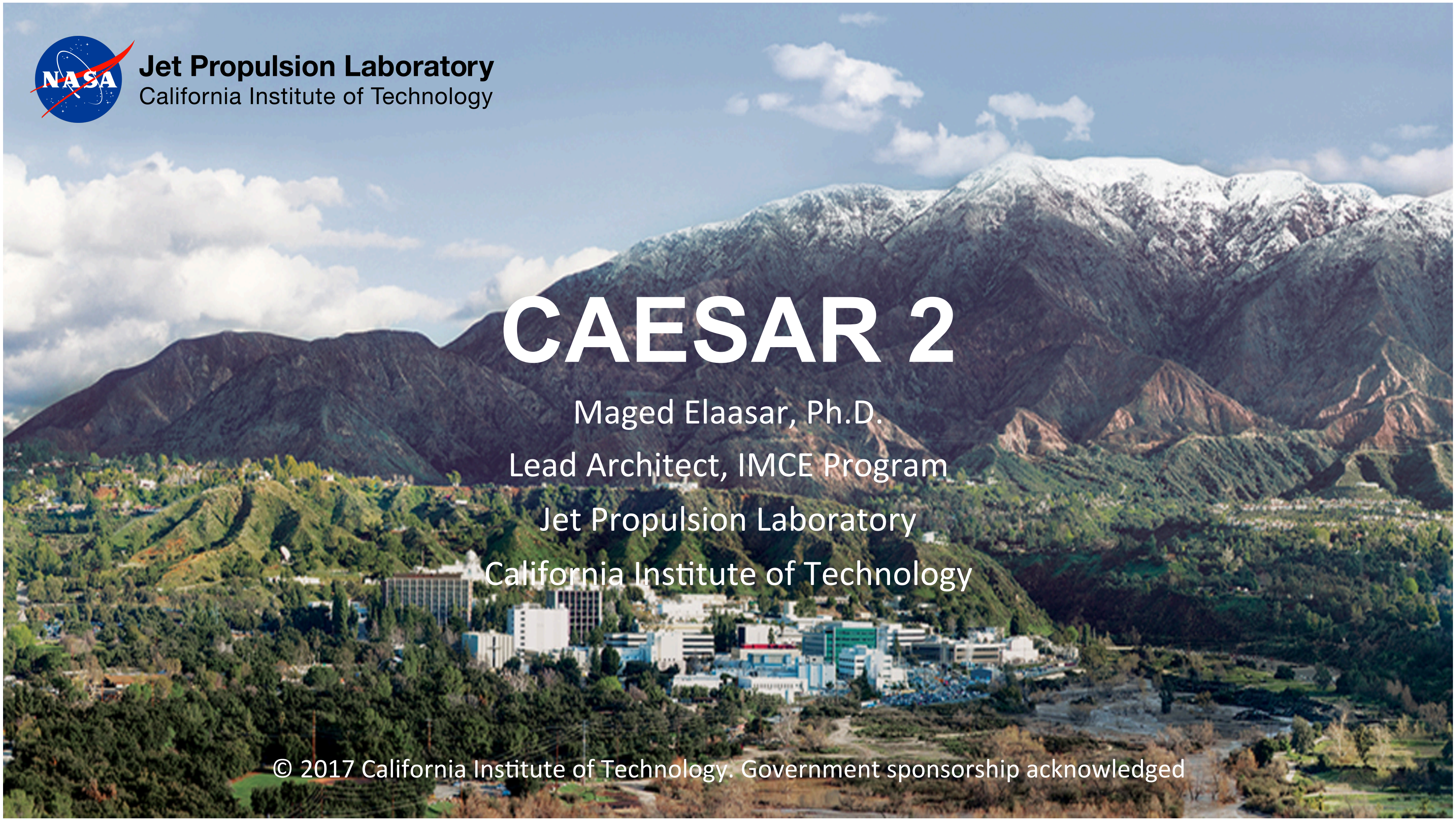


Jet Propulsion Laboratory
California Institute of Technology

CAESAR 2

Maged Elaasar, Ph.D.
Lead Architect, IMCE Program
Jet Propulsion Laboratory
California Institute of Technology

Disclaimer

Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

What is CAESAR?

Computer

Aided

Engineering for

System

Architecture design

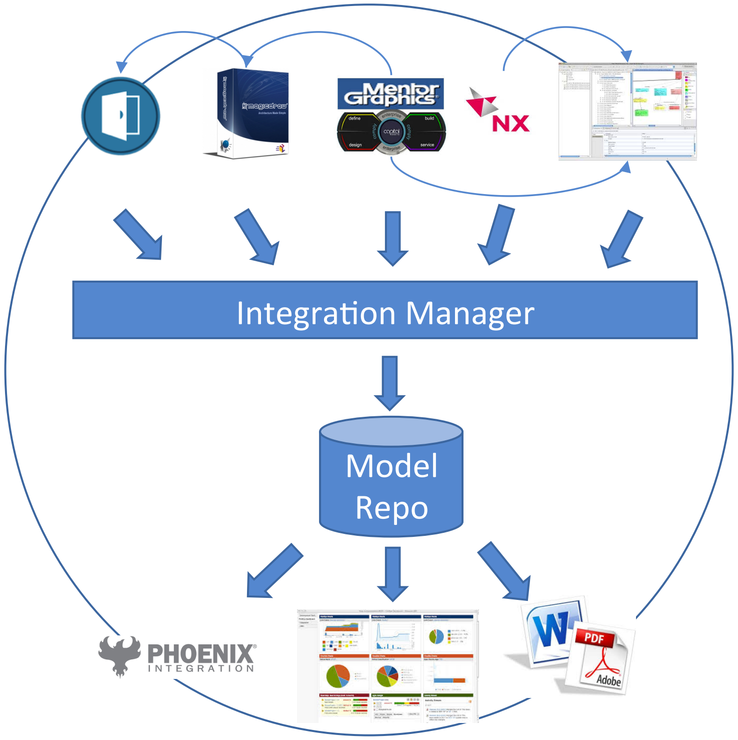
What CAESAR really is

- CAESAR is an integrated tool suite that supports the rigorous practice of model based systems engineering for space missions at JPL and enables
  - Development of system models that capture a technical baseline
  - Performing integrated analyses and trades using a system models
  - Generating systems engineering products, such as technical resources report
  - Conducting collaborative design sessions among systems engineers
  - Performing audits to ensure the integrity of the system design
  - Proposing and evaluating engineering changes and preserving change histories

Task motivation and objectives

- **Motivation**
  - In previous years, IMCE funded individual focused R&D tasks in the area of MBSE to produce prototype capabilities in < 1 year increments. That resulted in a lot of reports and proofs of concepts, but not a lot of integrated capabilities, not to mention ones that have any long-term support.
- **Objectives**
  - In FY17 the strategy was reorganized to put a significant part of the investment into a development team, an architecture, and focused development process intended to produce integrated and supported MBSE capabilities. We focus mainly on the architecture and process. The new process replaces the old process. There was no pre-existing architecture to speak of.
  - Our process is to work closely with process owners in the line and other domain experts to develop not only the tools, but also the SE methodology to use them effectively in a project. Rather than taking a breadth-first approach, we focus on specific use cases and take a deep look into the tools, process, and whole lifecycle of the information.
  - To accomplish this, IMCE will partner with CAE and the rest of the SE community who choose the COTS tools they want to use. We are try to integrate the information managed in those tools into a common information model. IMCE will work with flight projects, and hope that this new process will provide a vehicle, through which we can harvest some of the project-developed capabilities into institutionally-supported capabilities. We also hope to work with outside partners at other aerospace companies on common ontologies, and have open-sourced the ontologies and ontological analysis tools to support that.

The systems engineering journey at JPL


Integration Manager

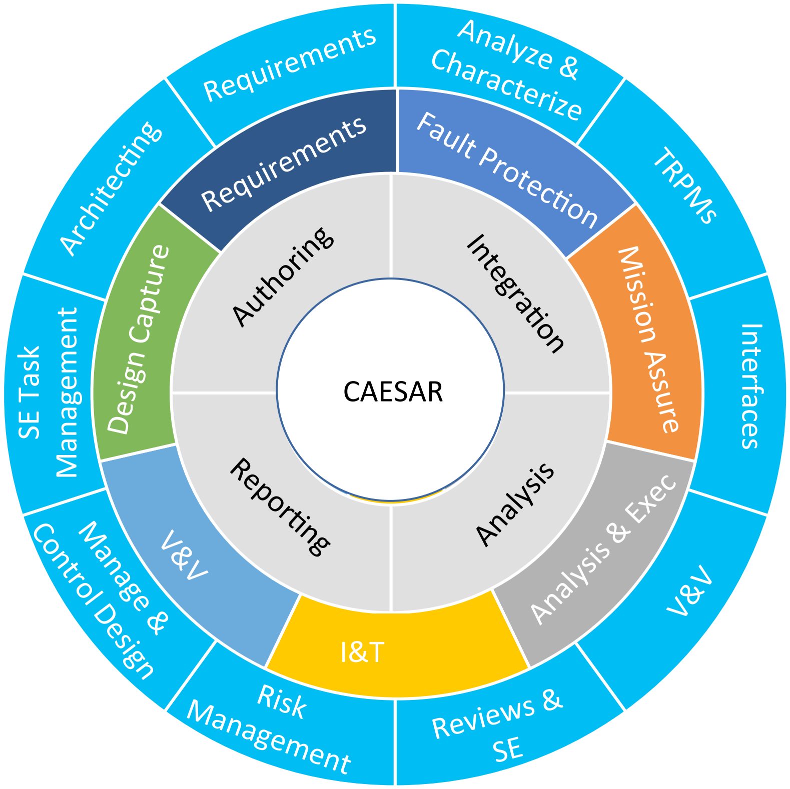
Model Repo

Document-based → Model-based → Rigorous, integrated, & streamlined model-based

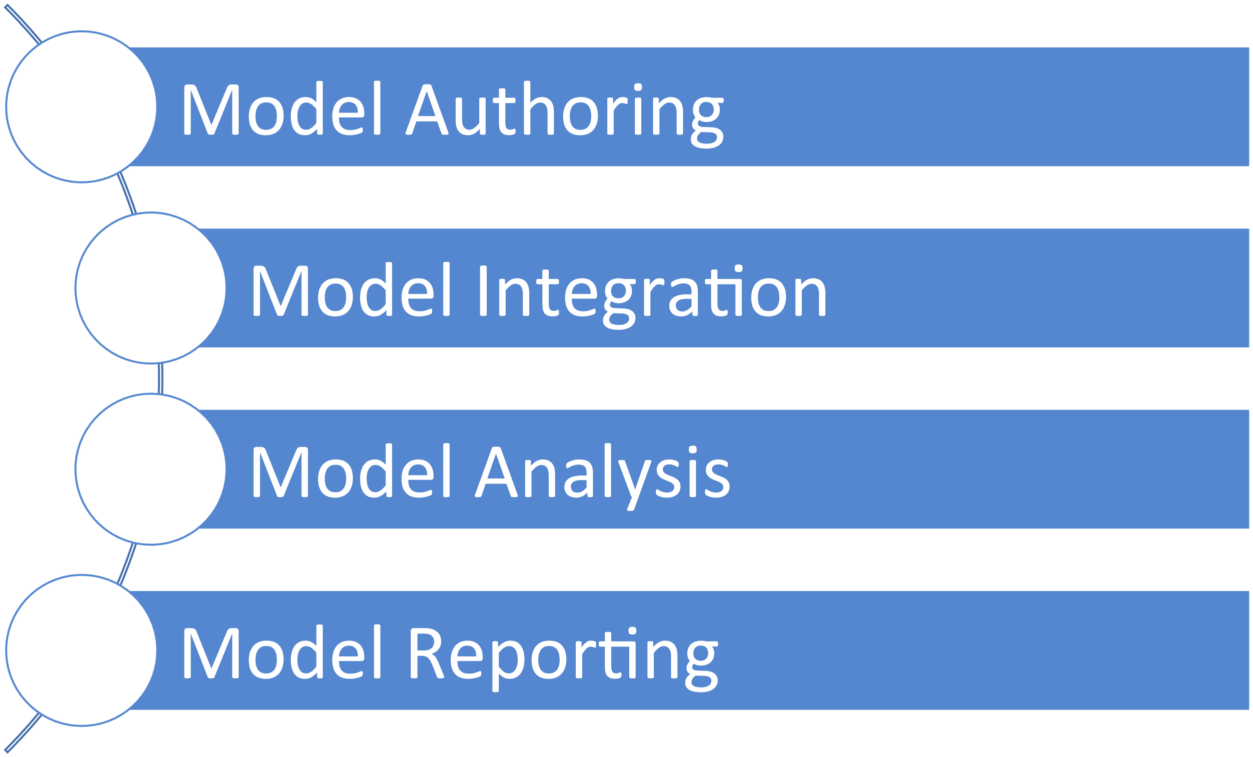
Business summary

Scope of work and vision

- CAESAR will provide an **infrastructure** that supports general MBSE capabilities like model authoring, integration, analysis and reporting
- CAESAR will provide **discipline**-specific capabilities on top of its infrastructure to support the various disciplines of systems engineering
- CAEAR will streamline the practice of the 10 JPL systems engineering **functions** by leveraging the discipline-specific capabilities



Stakeholder concern areas



Model Authoring

Model Integration

Model Analysis

Model Reporting

Model authoring concerns

- No adequate support for domain-specific vocabulary when using system modeling tools (e.g., MagicDraw)
- No adequate viewpoints to facilitate the expression of different system engineering concerns for different users/stakeholders
- No adequate tool support for methodologies for constructing system models
- No adequate reusable libraries of concepts and patterns to jump start the construction of system models
- No adequate support for common operations like model refactoring, transformation, validation, compare and merge, change management and tradeoff analysis

Model integration concerns

- System model is often fragmented into multiple silo tools with their own model repositories making it tedious to figure out a common baseline with governance and traceability
- Exchanging artifacts or creating relationships between artifacts in different tools is often not possible or does not work as expected
- Model fragments in different tools can easily become (and remain) inconsistent as integration between them occurs infrequently
- Model fragments from different tools often use disparate vocabularies and are expressed in different formats

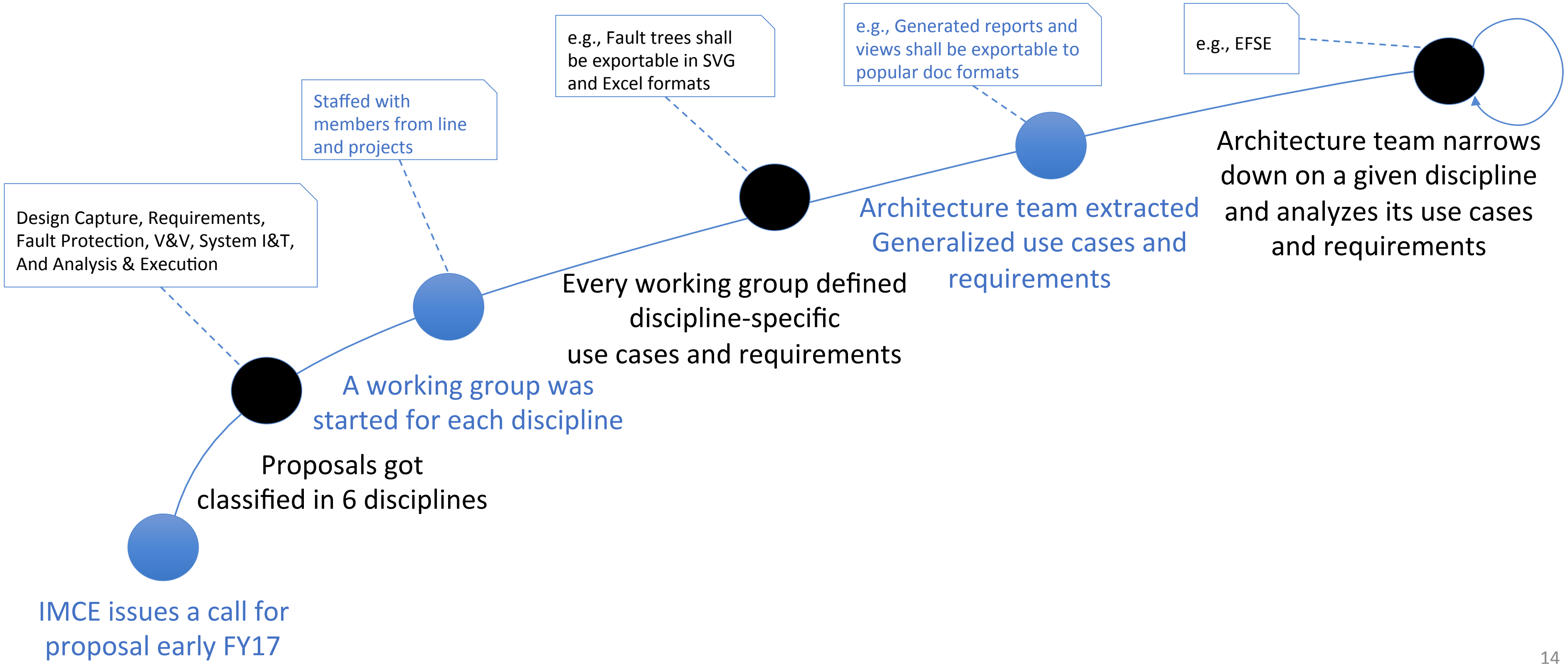
Model analysis concerns

- Model analysis are often expressed in a tool-specific way making it hard to support or switch to different tools
- Supported model vocabulary often has no well-defined semantics
- Analysis models are often not configuration managed and their results are not tracked and/or linked to relevant design decisions
- It is not easy to figure out when an analysis needs to be rerun, or how to run it automatically and/or incrementally
- It is often not easy to perform analysis on historical model baselines nor trending analysis across baselines
- It is often hard to run expensive analyses in a scalable fashion due to limitation on computing resources

Model reporting concerns

- It is often hard for stakeholders to follow up on the progress of a system model except at scheduled milestone reviews
- It is often hard to publish different views for a system model or organize them in dashboards/perspectives
- It is hard to produce consistent views from the system model due to lack of standard methodologies and/or view templates
- It is hard to allow stakeholders to browse, review, comment and approve the produced views
- It is hard to navigate from a view back to the authoring tool

Use cases & requirements process



IMCE issues a call for proposal early FY17

Proposals got classified in 6 disciplines

Design Capture, Requirements, Fault Protection, V&V, System I&T, And Analysis & Execution

A working group was started for each discipline

Staffed with members from line and projects

Every working group defined discipline-specific use cases and requirements

e.g., Fault trees shall be exportable in SVG and Excel formats

Architecture team extracted Generalized use cases and requirements

e.g., Generated reports and views shall be exportable to popular doc formats

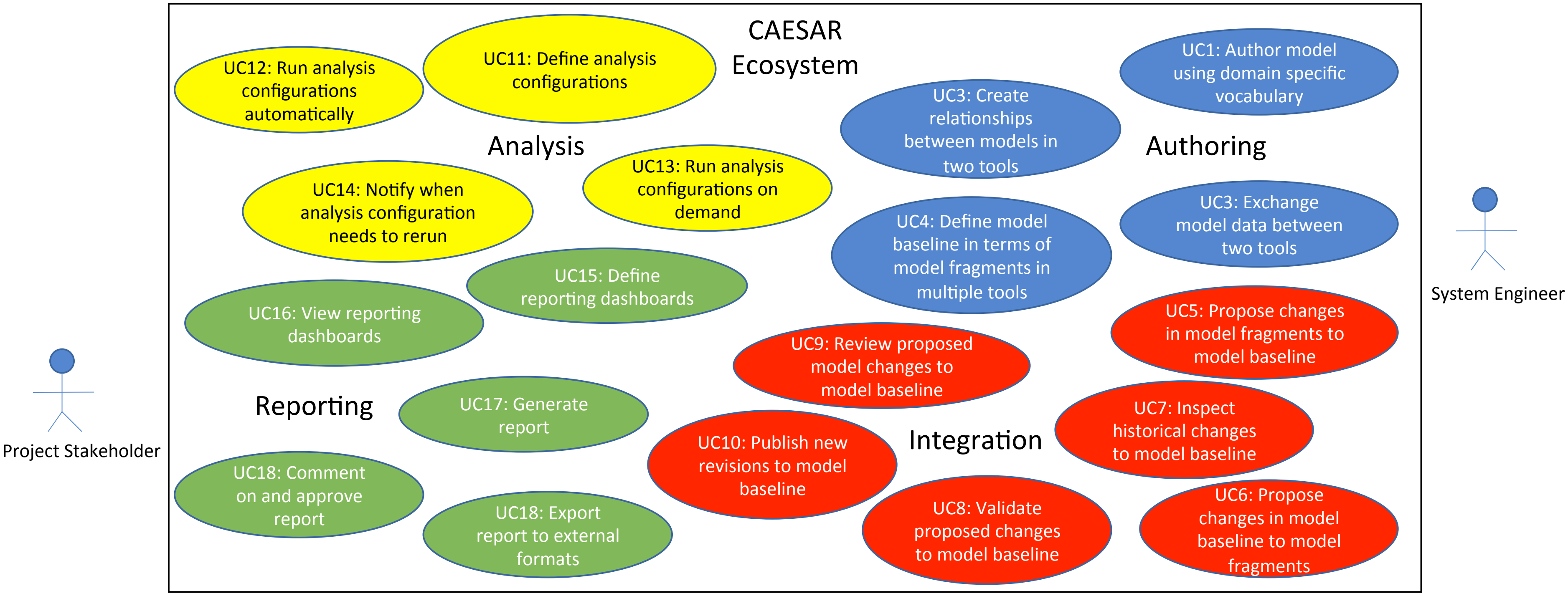
Architecture team narrows down on a given discipline and analyzes its use cases and requirements

e.g., EFSE

Use cases and requirements levels

- We organize use cases and requirements in levels
  - L1 : General
    - L2: General
      - L3: Discipline
        - L4: Discipline
          - L5: Application
            - L6: Application

General use cases


CAESAR Ecosystem
UC12: Run analysis configurations automatically
UC11: Define analysis configurations
UC1: Author model using domain specific vocabulary
UC3: Create relationships between models in two tools
Analysis
Authoring
UC14: Notify when analysis configuration needs to rerun
UC13: Run analysis configurations on demand
UC4: Define model baseline in terms of model fragments in multiple tools
UC3: Exchange model data between two tools
System Engineer
UC15: Define reporting dashboards
UC16: View reporting dashboards
UC5: Propose changes in model fragments to model baseline
UC9: Review proposed model changes to model baseline
Project Stakeholder
Reporting
UC17: Generate report
UC7: Inspect historical changes to model baseline
UC10: Publish new revisions to model baseline
Integration
UC18: Comment on and approve report
UC18: Export report to external formats
UC8: Validate proposed changes to model baseline
UC6: Propose changes in model baseline to model fragments

L1 requirements

- **REQ 1**: CAESAR shall facilitate authoring the system model using domain-specific vocabulary and viewpoints and through a set of federated SE tools
- **REQ 2**: CAESAR shall facilitate configuration management of system model baselines and streamline the processes of change proposal to them
- **REQ 3**: CAESAR shall facilitate continuous integration of system model baselines by supporting the processes of change review and validation
- **REQ 4**: CAESAR shall facilitate analysis of system model baselines by making them easily accessible to a variety of analysis tools and platforms
- **REQ 5**: CAESAR shall facilitate reporting on system model baselines, including creating visualizations, documents and dashboards
- **REQ 6**: CAESAR shall support cross cutting concerns with respect to deployment, operation and access control consistently across all subsystems

L2 requirements on model authoring

- **REQ 1.1**: CAESAR shall allow model authoring using domain-specific vocabularies with well-defined syntax and semantics
- **REQ 1.2**: CAESAR shall enable model authoring using a select set of over-the-shelf OTS tools
- **REQ 1.3**: CAESAR shall provide custom model authoring tools to fill functional gaps or address usability or scalability concerns with existing OTS tools
- **REQ 1.4**: CAESAR shall facilitate data linking, exchange and synchronization between a federated set of model authoring tools
- **REQ 1.5**: CAESAR shall provide model libraries and templates to jump start the modeling effort for new projects

L2 requirements on model management

- **REQ 2.1**: CAESAR shall create system model baselines based on defined configurations of model fragments in federated model authoring tools
- **REQ 2.2**: CAESAR shall provide interfaces to get and update model fragments in federated tools using a tool-neutral common vocabulary
- **REQ 2.3**: CAESAR shall enable associating revisions of a system model baseline to requested changes and/or milestones
- **REQ 2.4**: CAESAR shall support provenance of data in the system model baseline
- **REQ 2.5**: CAESAR shall facilitate the process of change proposal to a given system model baseline and apply the proposed changes to model fragments in the authoring tools

L2 requirements on model integration

- **REQ 3.1**: CAESAR shall continuously integrate the system model baseline by validating proposed changes to any model fragment in the baseline
- **REQ 3.2**: CAESAR shall report on detected validation errors in system model baseline in a way that enables understanding of error nature and scope
- **REQ 3.3**: CAESAR shall enable review, analysis and approval of change proposal to system model baselines by authorized project personnel

L2 requirements on model analysis

- **REQ 4.1**: CAESAR shall make both latest and historical system model baselines accessible to analysis tools
- **REQ 4.2**: CAESAR shall make system model baselines readable by analysis tools through a set of vocabulary-based API
- **REQ 4.3**: CAESAR shall allow the definition of analysis configurations that analyzes (one or more revisions of) the system model baseline
- **REQ 4.4**: CAESAR shall enable analysis to run either automatically or on demand
- **REQ 4.5**: CAESAR shall detect whether an analysis needs to re-run based on changes to the system model baseline and notify watching stakeholders

L2 requirements on model reporting

- **REQ 5.1**: CAESAR shall provide a unified portal for reporting on the status of the system model baseline for a given project
- **REQ 5.2**: CAESAR shall allow role-based access to the reporting portal to enable configuring what users can see and do
- **REQ 5.3**: CAESAR shall allow visualization of analysis results using different kinds of UI widgets
- **REQ 5.4**: CAESAR shall allow organizing visualization widgets in pre-defined or custom-defined dashboards
- **REQ 5.5**: CAESAR shall allow exporting of analysis results in popular external formats

L2 requirements on cross cutting concerns

- **REQ 6.1**: CAESAR shall support multi-tenancy (multiple projects to coexist in the same deployment)
- **REQ 6.2**: CAESAR shall control access to system model data consistent with institutional mandates (e.g., ITAR)
- **REQ 6.3**: CAESAR shall support role-based access to its different subsystems

Architecture description

Architectural principles

- The architecture should be implementable over time in a modular and incremental fashion
- The architecture should consider leveraging or extending OTS tools before developing new ones
- The architecture should integrate OTS tools in a way that minimizes the impact of future revisions
- The architecture should build new tools on proven technologies with growing communities
- The architecture should prioritize using open standards and open source technologies
- The architecture will incubate designs and implementations until they prove to be useful, before committing to their long term support

Information architecture

- Model based system engineering (MBSE) is about managing information as models
- No single tool can effectively manage all information
- Managing information is done through federating multiple tools
- Tool federation focuses on exchange and analysis of information
- Information in each tools is collected in a common baseline for analysis
  - This allows managing the in-work SE information (not only gate products) consistently
- Information is constantly changing; but the baseline needs to evolve consistently
  - The change is happening on different timescales and frequencies
  - The consistency needs to be checked continuously and in an automated way
- Information needs to be published uniformly, consistently, and in a known place

Information as ontologies

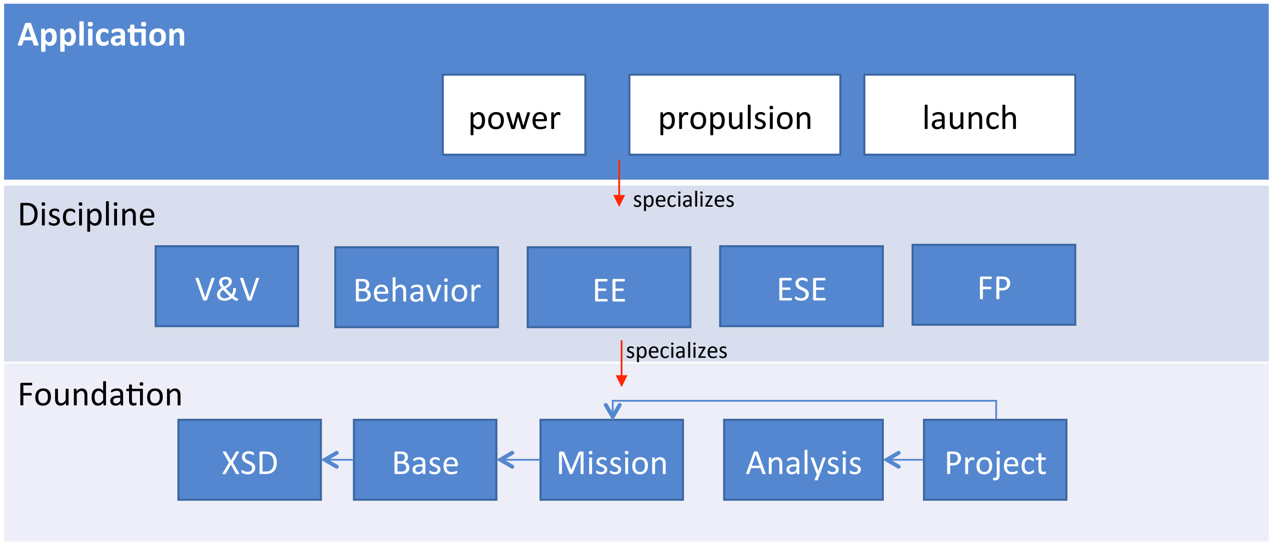
- Models can be represented as ontologies (a set of concepts with properties/relationships)
- An ontology can be of two kinds:
  - Terminology (T-box): defines a set of classes, relationships and properties
  - Description (A-box): define a set of instances of classes and relationships for analysis
- Defining terminologies should be the first step in information modeling
  - Defines the information we care about and agree on (like agreeing on interfaces)
  - Allows tools that conform to those terminologies to be switched in a plug and play fashion

Ontology Modeling Language

- CAESAR defines ontologies using the Ontology Modeling Language (OML)
- OML is a new domain-specific language for modeling ontologies
  - Abstract syntax is defined based on the W3C standard Web Ontology Language (OWL)
  - Concrete syntax is defined with a concise textual grammar and graphical notation
  - Semantics is defined based on Description Logic (DL)
- CAESAR introduces OML is to create correct-by-construction ontologies, facilitate working with ontologies, and address some of the limitations of OWL (e.g., consistent serialization)

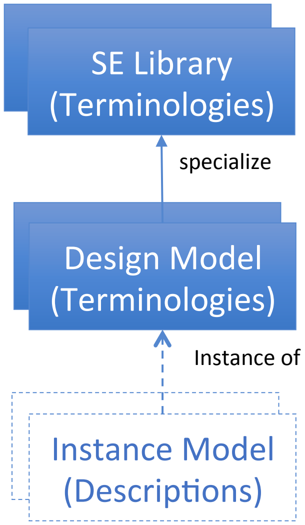
Systems engineering ontologies

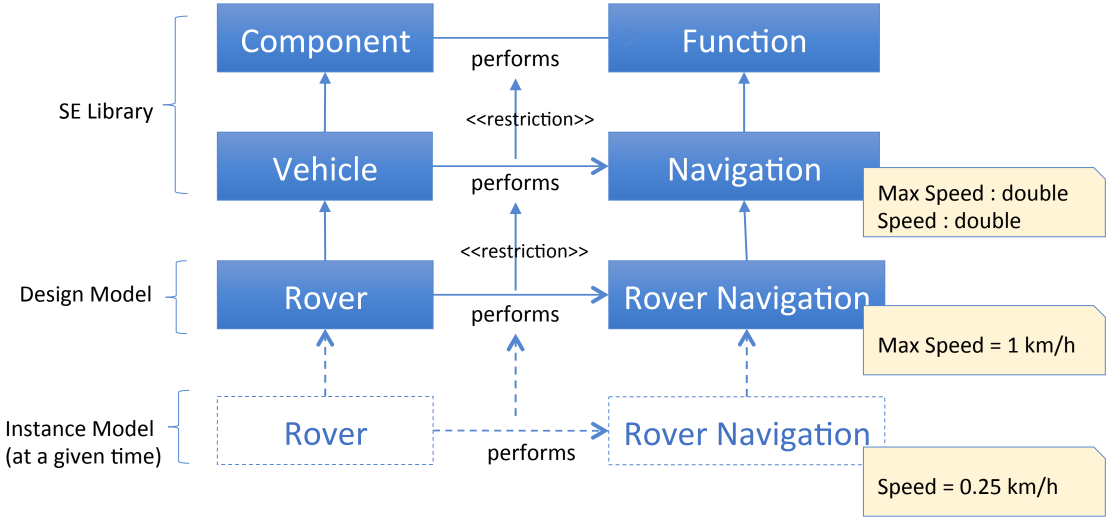
- CAESAR defines a library of terminologies for space mission systems engineering and organizes them in 3 layers



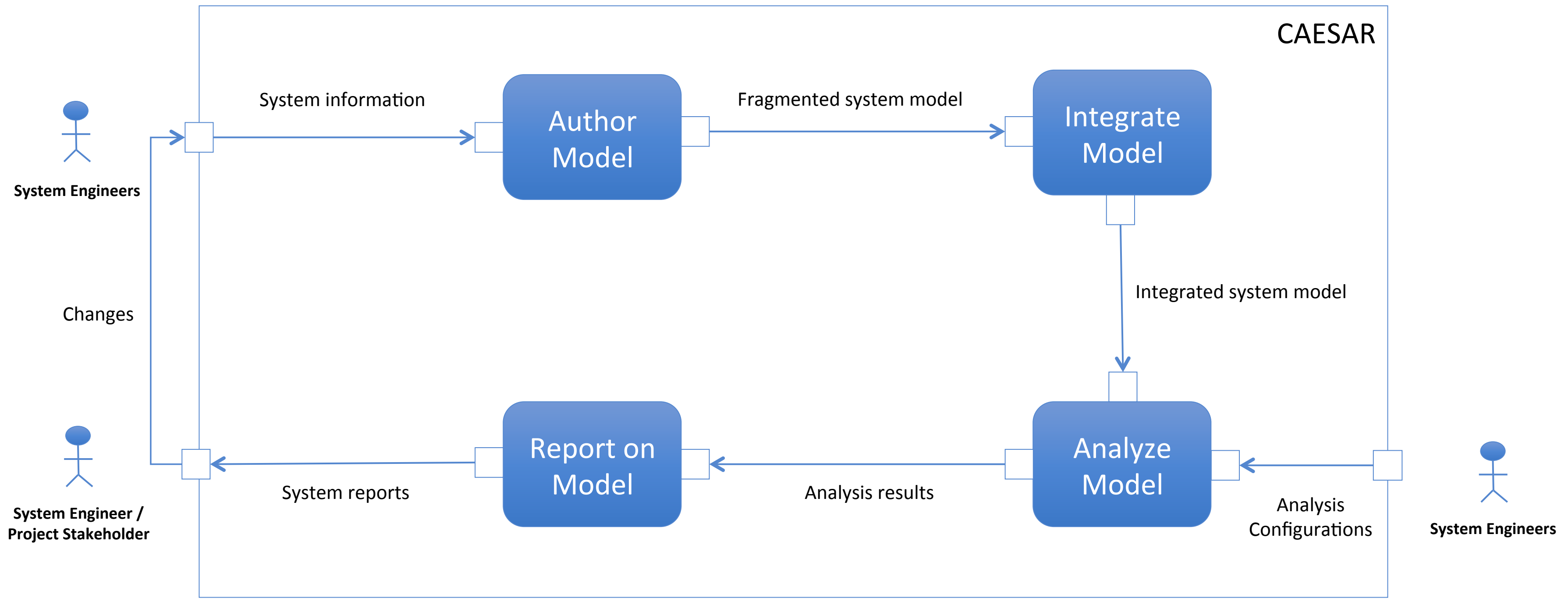
Project ontologies

- Projects specify design models as terminologies that specialize the library of SE terminologies
- Projects specify instance models for analysis (at a given time) as descriptions

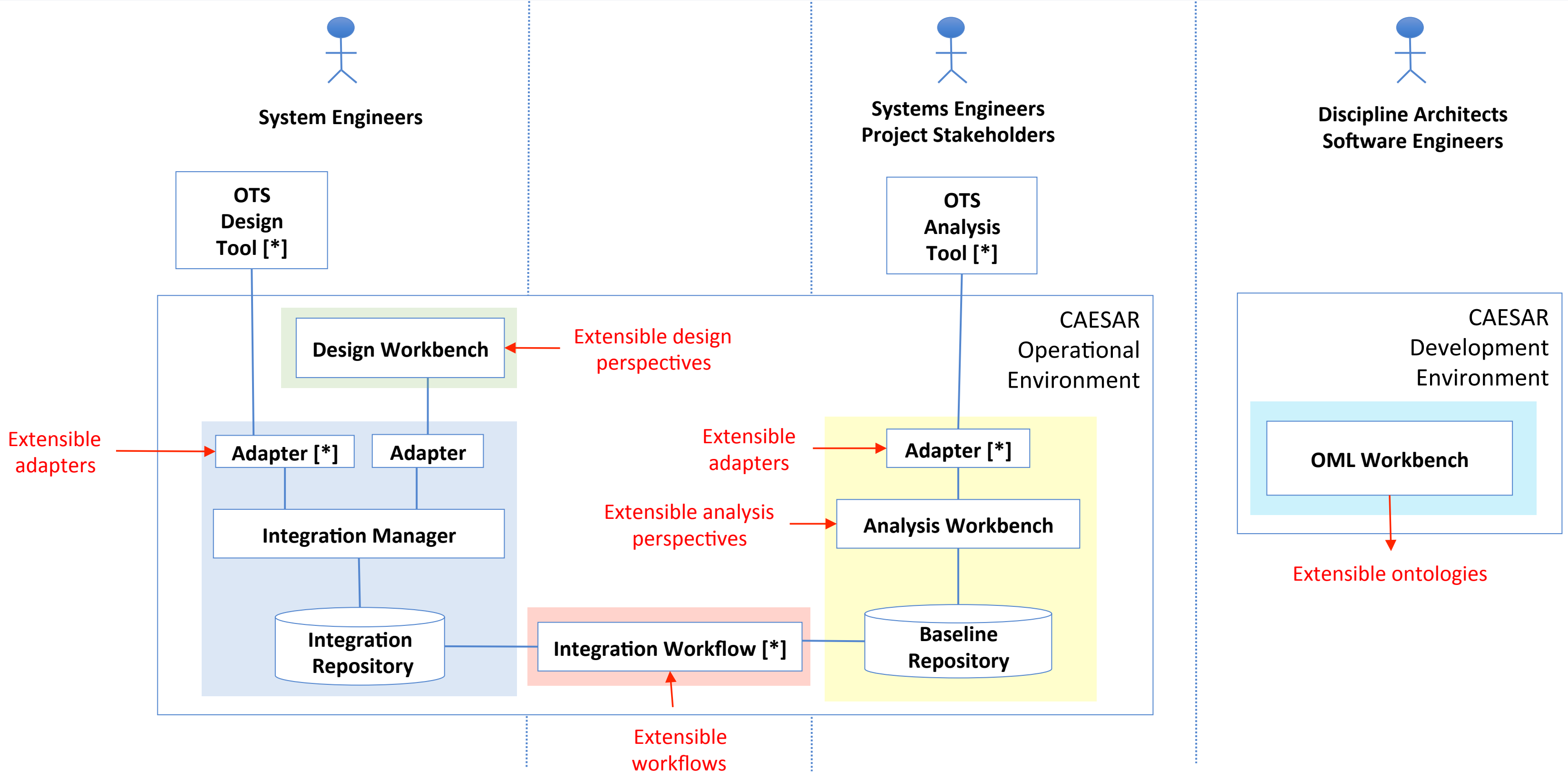

SE Library (Terminologies)
specialize
Design Model (Terminologies)
Instance of
Instance Model (Descriptions)



Process architecture

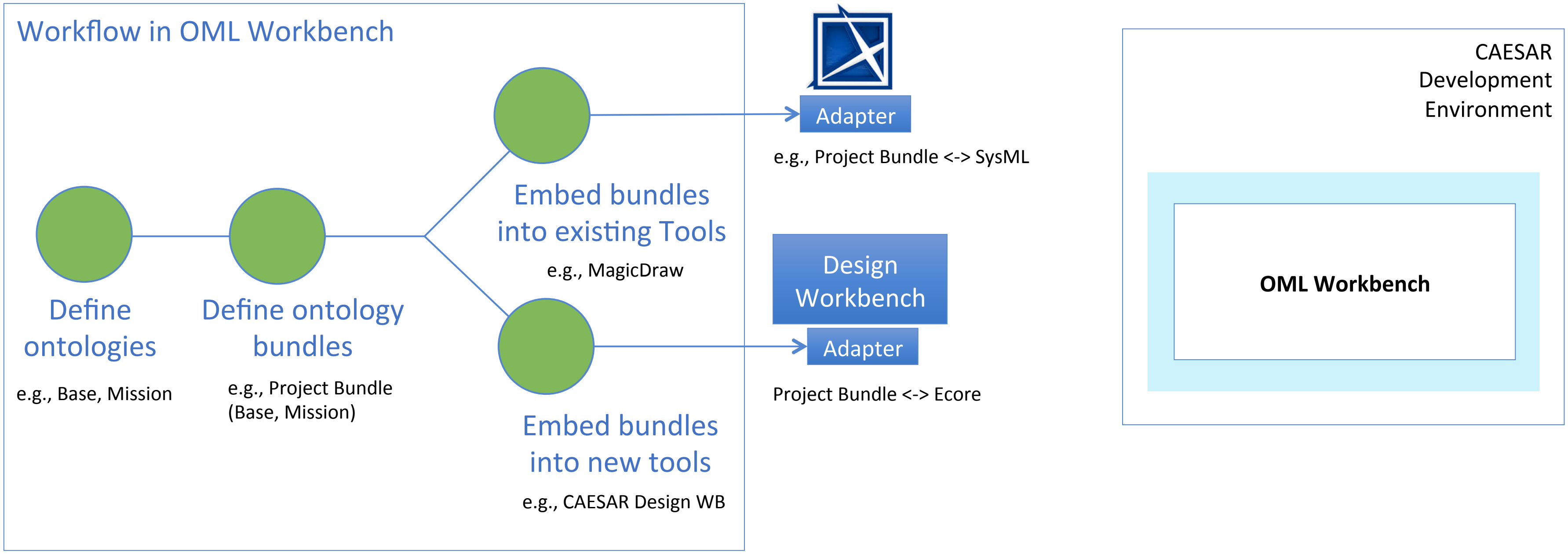


Software architecture



OML workbench

- Provide a workbench for the development and embedding of OML ontologies



Workflow in OML Workbench

Define ontologies

e.g., Base, Mission

Define ontology bundles

e.g., Project Bundle (Base, Mission)

Embed bundles into existing Tools

e.g., MagicDraw

Embed bundles into new tools

e.g., CAESAR Design WB

Adapter

e.g., Project Bundle <-> SysML

Design Workbench

Adapter

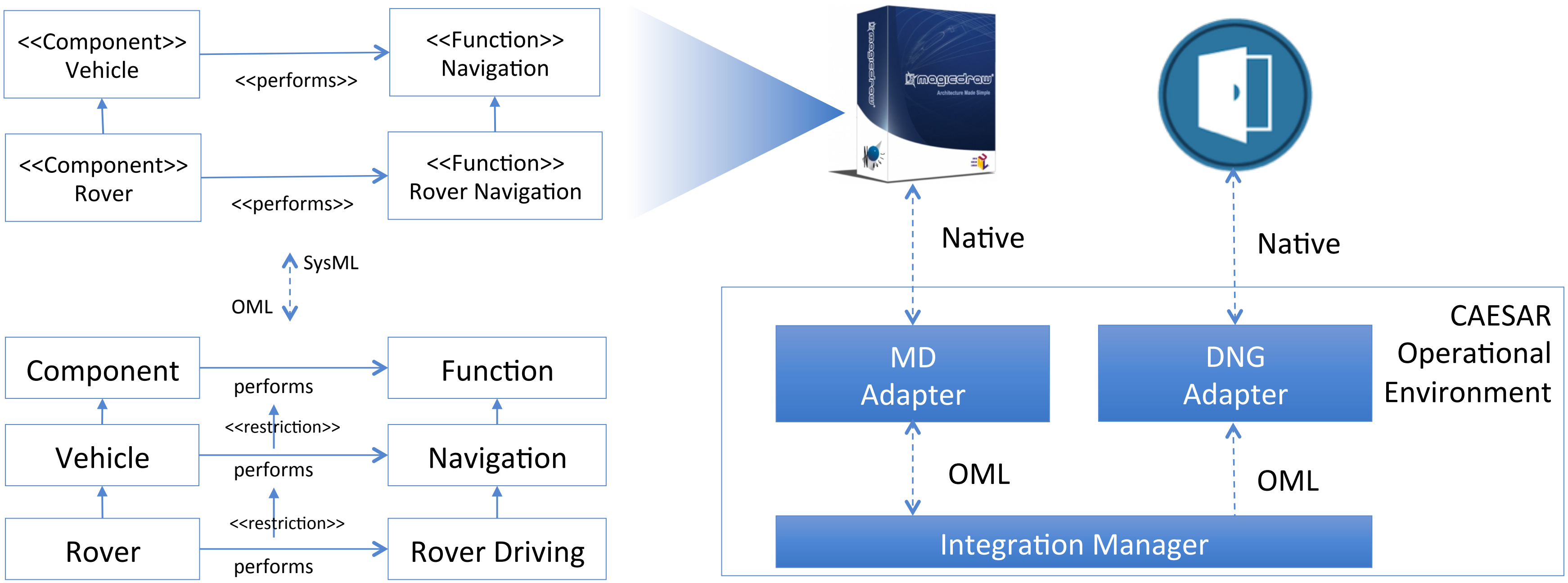
Project Bundle <-> Ecore

CAESAR Development Environment

OML Workbench

Adapters to existing design tools

- Provide adapters to existing design tools that map between the tools' native formats and OML



<<Component>> Vehicle

<<performs>>

<<Function>> Navigation

<<Component>> Rover

<<performs>>

<<Function>> Rover Navigation

SysML

OML

Component

performs

Function

<<restriction>>

Vehicle

performs

Navigation

<<restriction>>

Rover

performs

Rover Driving

Native

Native

CAESAR Operational Environment

MD Adapter

DNG Adapter

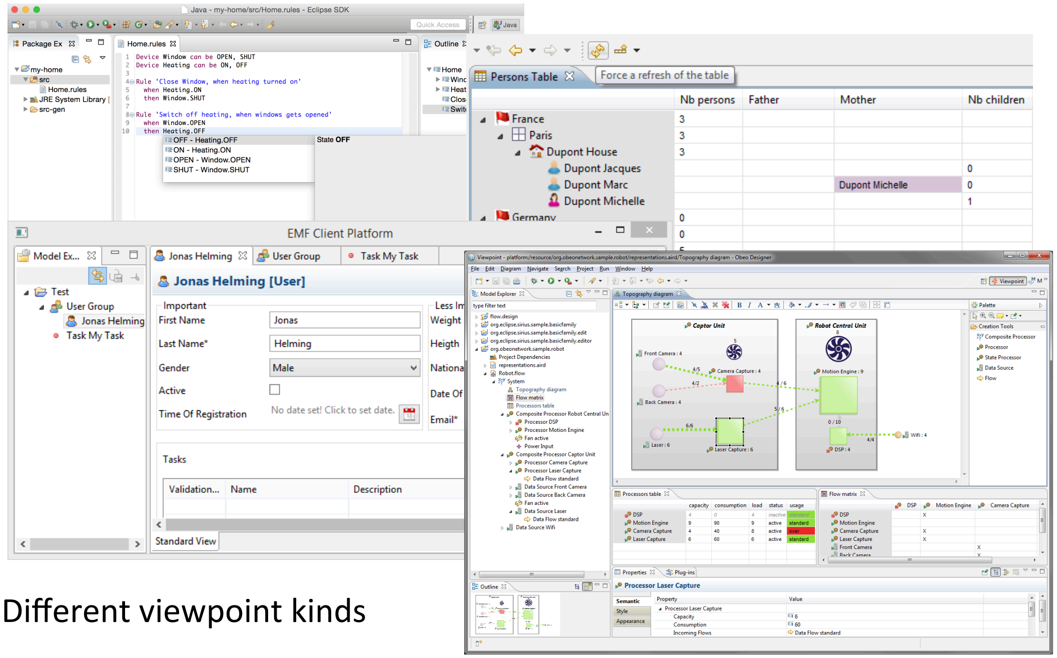
OML

OML

Integration Manager

Design workbench

- Provide a Design Workbench to implement missing or more usable domain-specific design perspectives



Java - my-home/src/Home.rules - Eclipse SDK

```
Device Window can be OPEN, SHUT
Device Heating can be ON, OFF

Rule 'Close Window, when heating turned on'
  when Heating.ON
  then Window.SHUT

Rule 'Switch off heating, when windows gets opened'
  when Window.OPEN
  then Heating.OFF
```

OFF - Heating.OFF
ON - Heating.ON
OPEN - Window.OPEN
SHUT - Window.SHUT

State **OFF**

Persons Table — Force a refresh of the table

	Nb persons	Father	Mother	Nb children
France	3			
Paris	3			
Dupont House	3			
Dupont Jacques				0
Dupont Marc			Dupont Michelle	0
Dupont Michelle				1
Germany	0			
	0			

EMF Client Platform

Jonas Helming [User]

Important

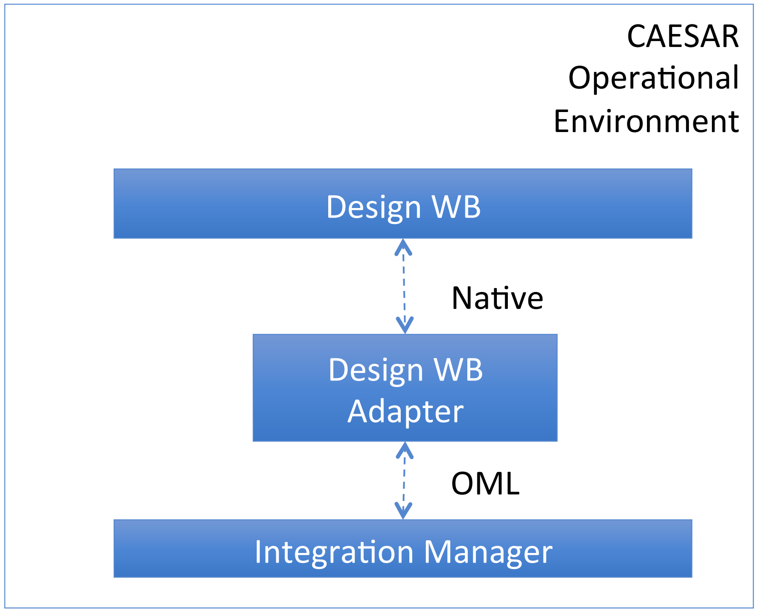
First Name	Jonas
Last Name*	Helming
Gender	Male
Active	□
Time Of Registration	No date set! Click to set date.

Tasks

Validation...	Name	Description

Viewpoint - platform:/resource/org.obeonetwork.sample.robot/representations.aird/Topography diagram - Obeo Designer

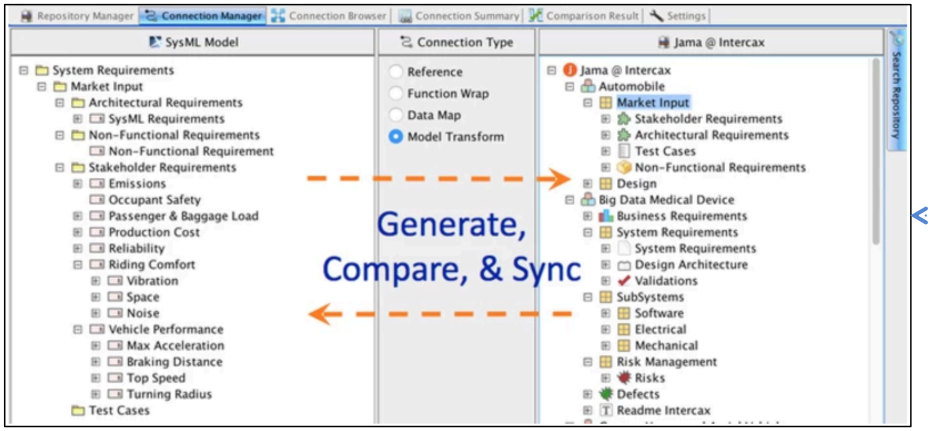
Different viewpoint kinds



CAESAR Operational Environment

Link manager(s)

- Provide tool(s) that support linking model fragments in SE tools
  - Creates relationships between model fragments in tools
  - Generate, compare and sync model fragments between tools

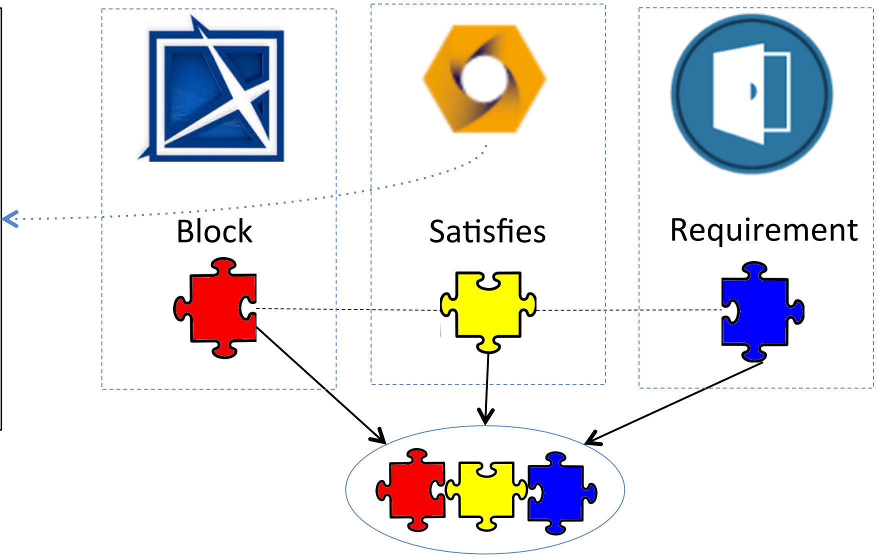


- System Requirements
  - Market Input
    - Architectural Requirements
      - SysML Requirements
    - Non-Functional Requirements
      - Non-Functional Requirement
    - Stakeholder Requirements
      - Emissions
      - Occupant Safety
      - Passenger & Baggage Load
      - Production Cost
      - Reliability
      - Riding Comfort
        - Vibration
        - Space
        - Noise
      - Vehicle Performance
        - Max Acceleration
        - Braking Distance
        - Top Speed
        - Turning Radius
    - Test Cases

○ Reference
○ Function Wrap
○ Data Map
● Model Transform

- Jama @ Intercax
  - Automobile
    - Market Input
      - Stakeholder Requirements
      - Architectural Requirements
      - Test Cases
      - Non-Functional Requirements
    - Design
  - Big Data Medical Device
    - Business Requirements
    - System Requirements
      - System Requirements
      - Design Architecture
      - Validations
    - SubSystems
      - Software
      - Electrical
      - Mechanical
    - Risk Management
      - Risks
    - Defects
    - Readme Intercax

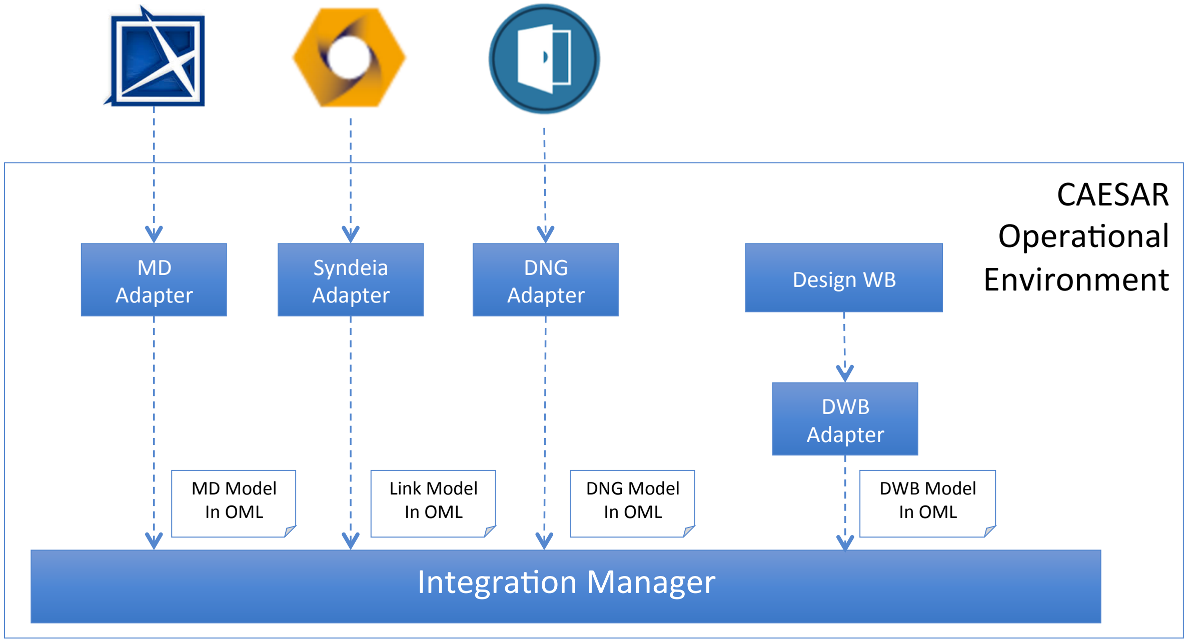
Link Manager (E.g., Intercax Syndeia)



Linked System Model

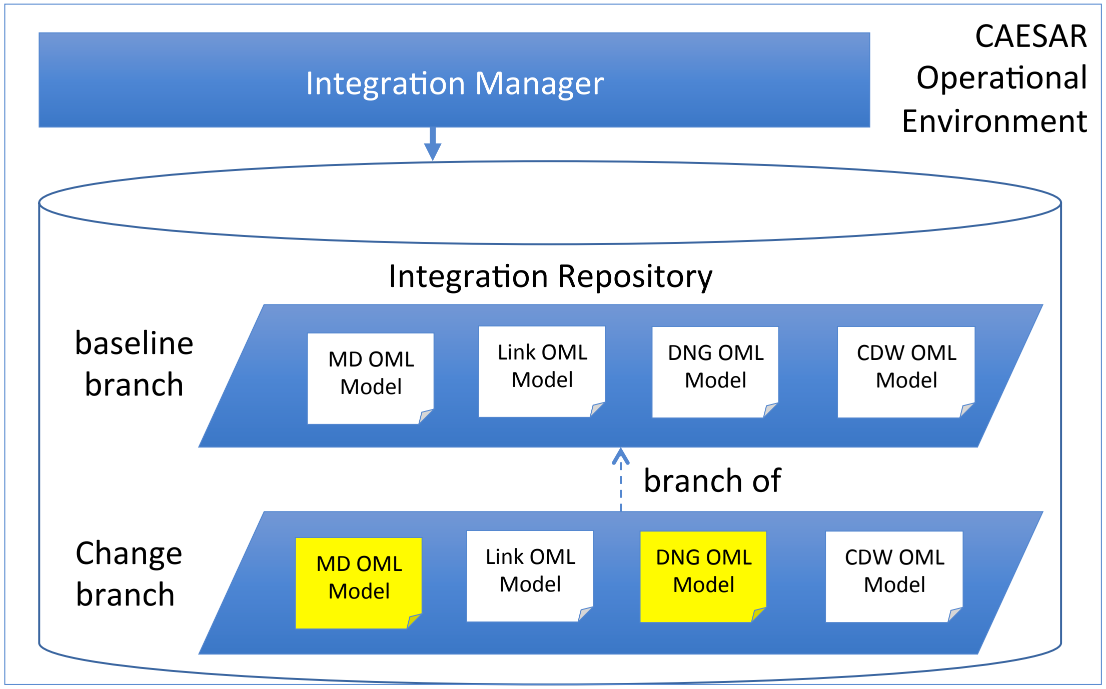
Integration manager

- Use Integration Manager to pull the (changed) system model fragments in OML from the different design tools



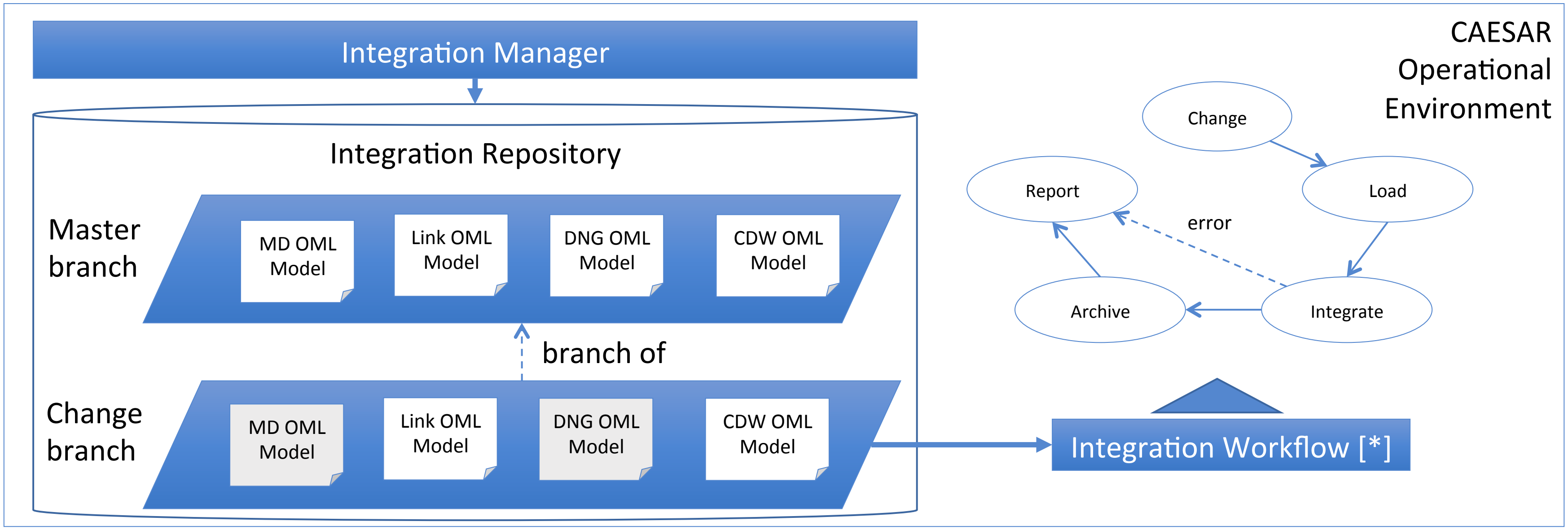
Integration repository

- System model baseline is maintained in a root branch in the integration repository
- Upon performing an integration, the Integration Manager creates a new change branch to hold the changed fragments of the system model
- Integration Manager then requests a merge of the change branch into baseline branch



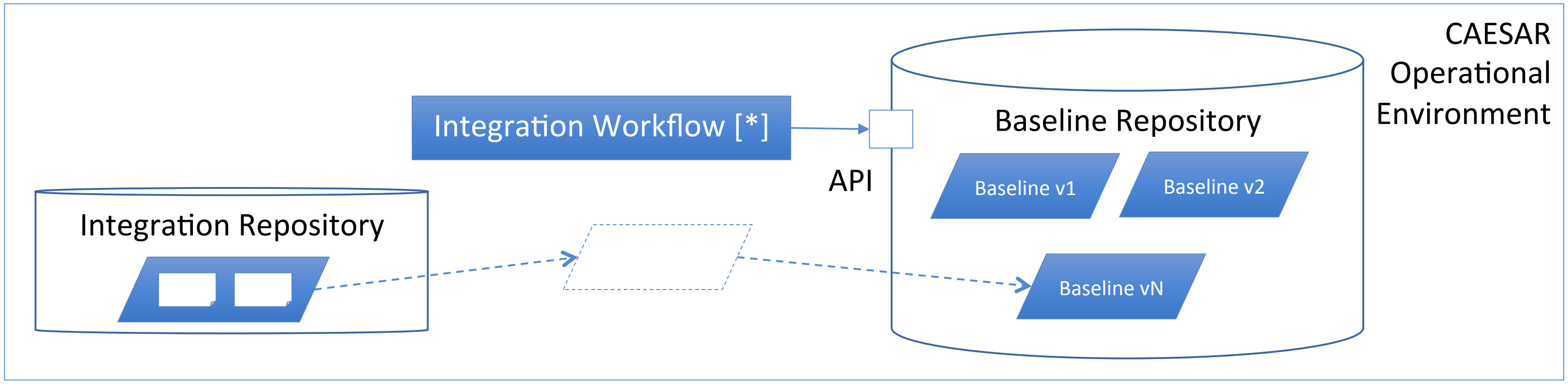
Continuous integration

- A CI job gets triggered by the merge request, causing integration workflows to run and report integration errors (if any)
  - The workflows are configurable per projects in Integration Manager



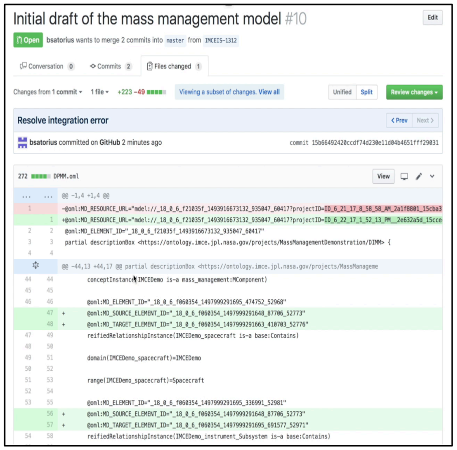
Integration workflows

- The integration workflows first read the OML data from an Integration Repository’s branch and load it as a baseline revision in the Baseline Repository
- Baseline Repository is a scalable database that manages historic baselines (with ids like <project>.<branch>.<timestamp>) and exposes ontology-based API
- Integration workflows query the baseline data, run analysis, and store results back in the baseline (as derived data)



Baseline change inspection

- A review request is sent by email to a project leader who can choose to inspect the change deltas in the proposal branch
- The deltas may be inspected as text diffs or structural diffs
- The reviewer may then request further changes to the model by leaving comments in the review request
- The reviewer may eventually accept the changes and merge them to the integration branch



Initial draft of the mass management model #10

bsatorius wants to merge 2 commits into master from IMCEIS-1312

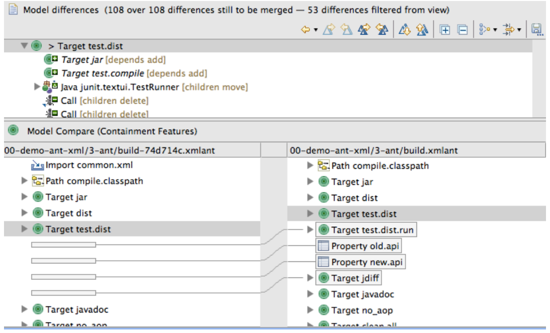
Resolve integration error

bsatorius committed on GitHub 2 minutes ago commit 15b66492420ccdf74d230e11d04b4651fff29031

272 DPMM.oml

```
@@ -1,4 +1,4 @@
-@oml:MD_RESOURCE_URL="mdel://_18_0_6_f21035f_1493916673132_935047_60417?projectID=ID_6_21_17_8_58_58_AM_2a1f8801_15cba3
+@oml:MD_RESOURCE_URL="mdel://_18_0_6_f21035f_1493916673132_935047_60417?projectID=ID_6_22_17_1_52_13_PM__2e632a5d_15cce
 @oml:MD_ELEMENT_ID="_18_0_6_f21035f_1493916673132_935047_60417"
 partial descriptionBox <https://ontology.imce.jpl.nasa.gov/projects/MassManagementDemonstration/DIMM> {

@@ -44,13 +44,17 @@ partial descriptionBox <https://ontology.imce.jpl.nasa.gov/projects/MassManageme
         conceptInstance(IMCEDemo is-a mass_management:MComponent)

         @oml:MD_ELEMENT_ID="_18_0_6_f060354_1497999291695_474752_52968"
+        @oml:MD_SOURCE_ELEMENT_ID="_18_0_6_f060354_1497999291648_87706_52773"
+        @oml:MD_TARGET_ELEMENT_ID="_18_0_6_f060354_1497999291663_410703_52776"
         reifiedRelationshipInstance(IMCEDemo_spacecraft is-a base:Contains)

         domain(IMCEDemo_spacecraft)=IMCEDemo

         range(IMCEDemo_spacecraft)=Spacecraft

         @oml:MD_ELEMENT_ID="_18_0_6_f060354_1497999291695_336991_52981"
+        @oml:MD_SOURCE_ELEMENT_ID="_18_0_6_f060354_1497999291648_87706_52773"
+        @oml:MD_TARGET_ELEMENT_ID="_18_0_6_f060354_1497999291695_691577_52971"
         reifiedRelationshipInstance(IMCEDemo_instrument_Subsystem is-a base:Contains)
```

Textual Diff



Model differences (108 over 108 differences still to be merged — 53 differences filtered from view)

> Target test.dist
Target jar [depends add]
Target test.compile [depends add]
Java junit.textui.TestRunner [children move]
Call [children delete]
Call [children delete]

Model Compare (Containment Features)

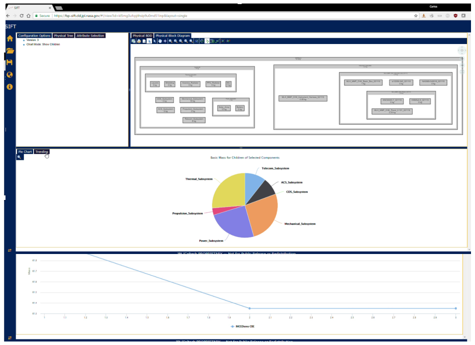
00-demo-ant-xml/3-ant/build-74d714c.xmlant
Import common.xml
Path compile.classpath
Target jar
Target dist
Target test.dist
Target javadoc
Target no_aop

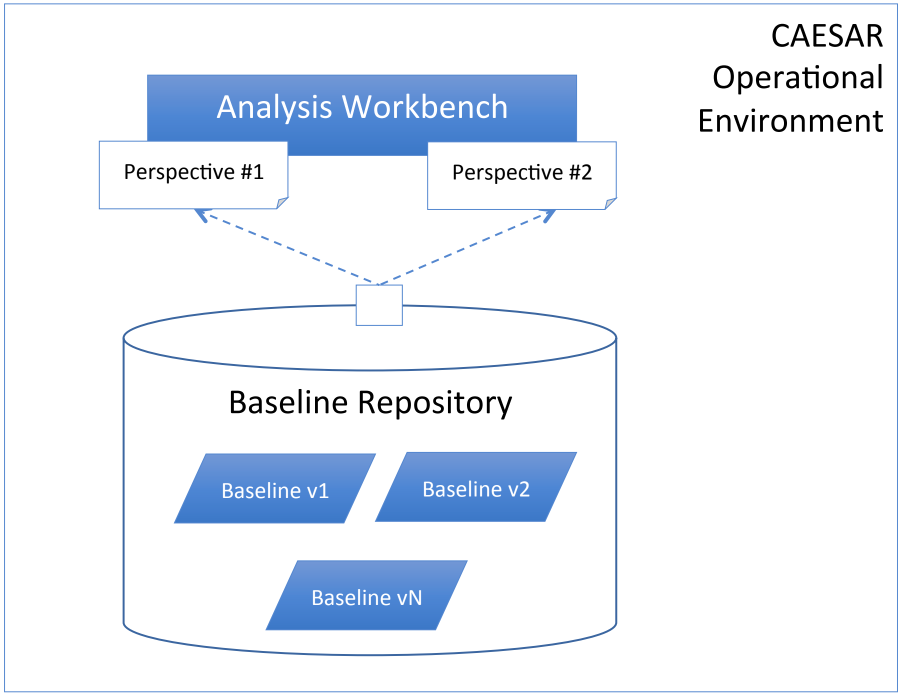
00-demo-ant-xml/3-ant/build.xmlant
Path compile.classpath
Target jar
Target dist
Target test.dist
Target test.dist.run
Property old.api
Property new.api
Target jdiff
Target javadoc
Target no_aop
Target clean_all

Structural Diff

Baseline analysis inspection

- Analysis Workbench allows users to inspect the results of analyzing a baseline revision (or a set of them) in the Baseline Repository through a set of domain-specific perspectives
- Each perspective contains widgets that reports on certain aspects of the system model baseline
- Perspectives will be customizable




CAESAR Operational Environment

Analysis Workbench

Perspective #1

Perspective #2

Baseline Repository

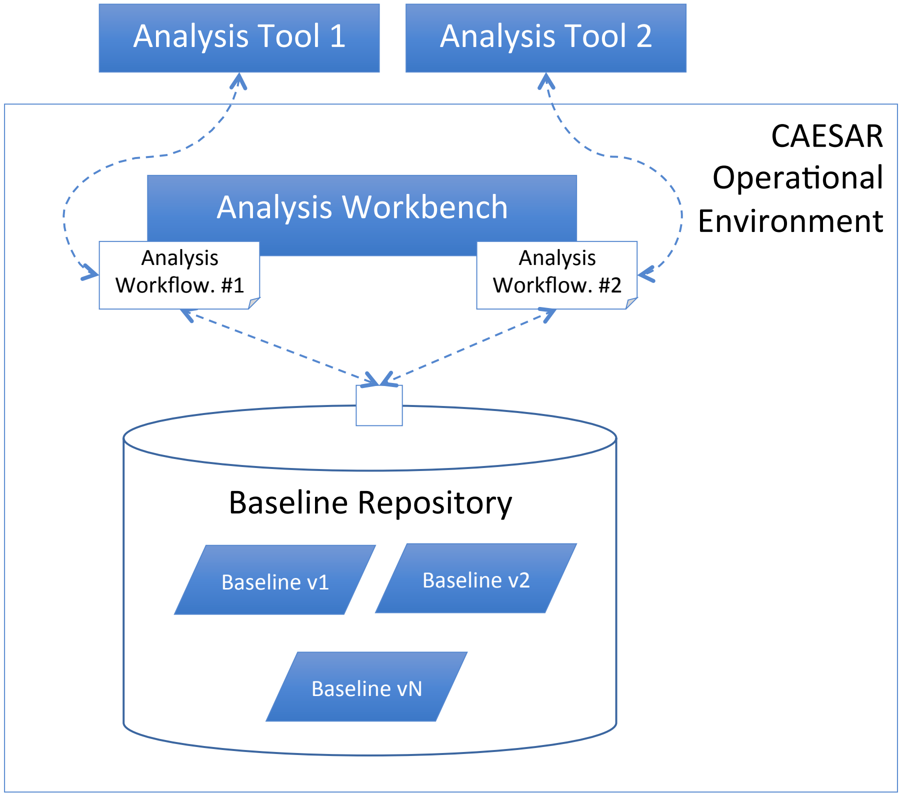
Baseline v1

Baseline v2

Baseline vN

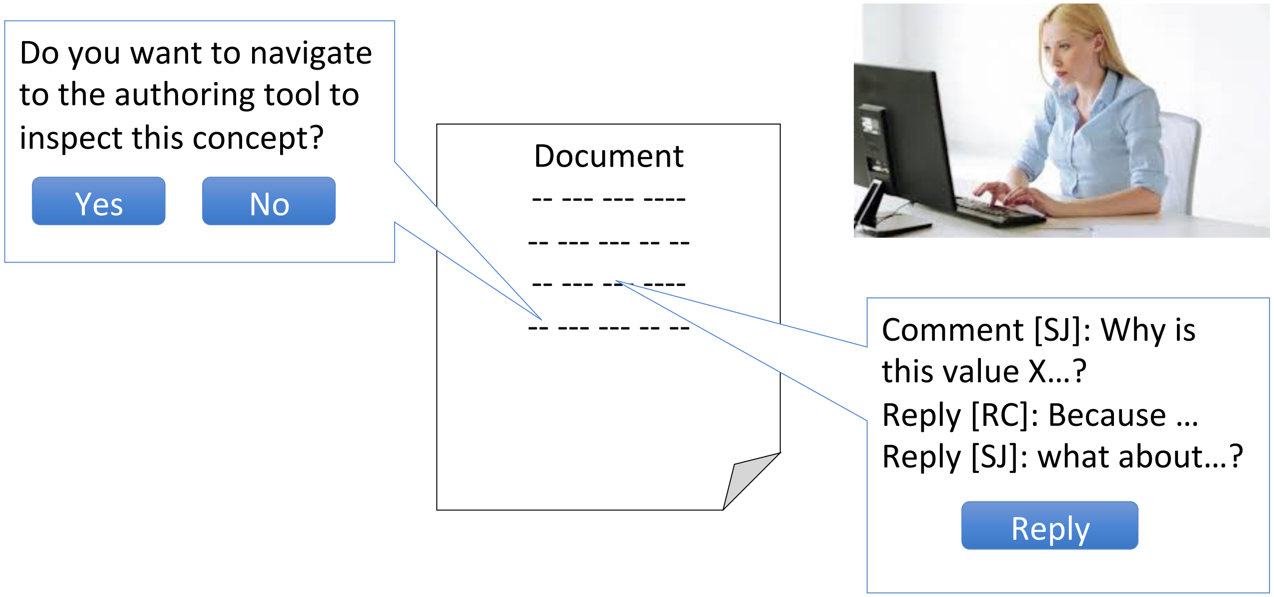
Analysis workflows

- The analysis workbench also supports the ability to define and register analysis workflows
- Analysis workflows read the baseline data from the Baseline Repository, process them and report on them
- Integration workflow is a kind of analysis workflow that is run automatically upon integration, as opposed to on-demand within the Analysis Workbench



Gate product inspection

- Analysis Workbench will allow users to generate domain-specific gate products, review them, comment on them, and finally approve them



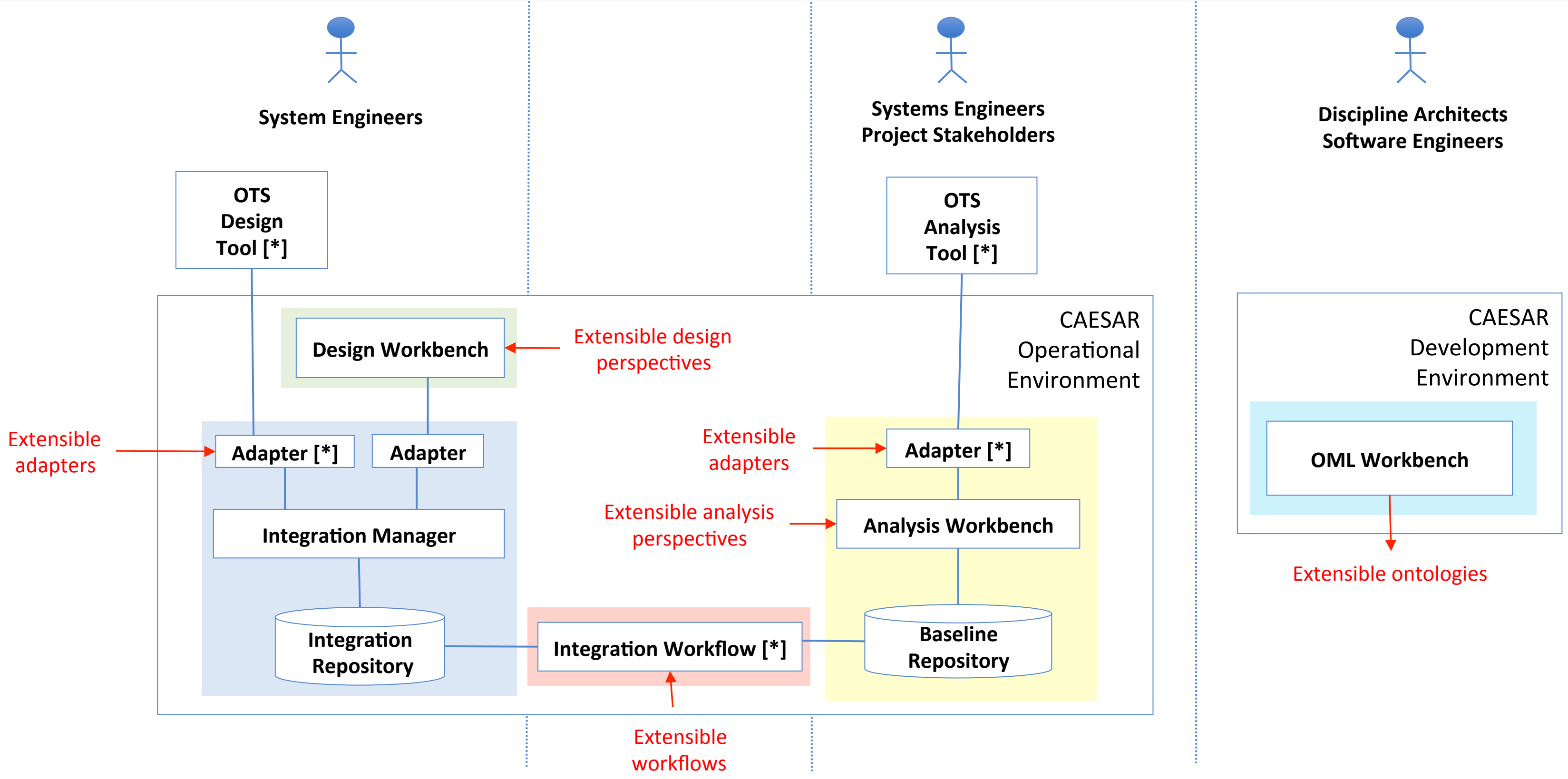
Do you want to navigate to the authoring tool to inspect this concept?

Comment [SJ]: Why is this value X…?

Reply [RC]: Because …

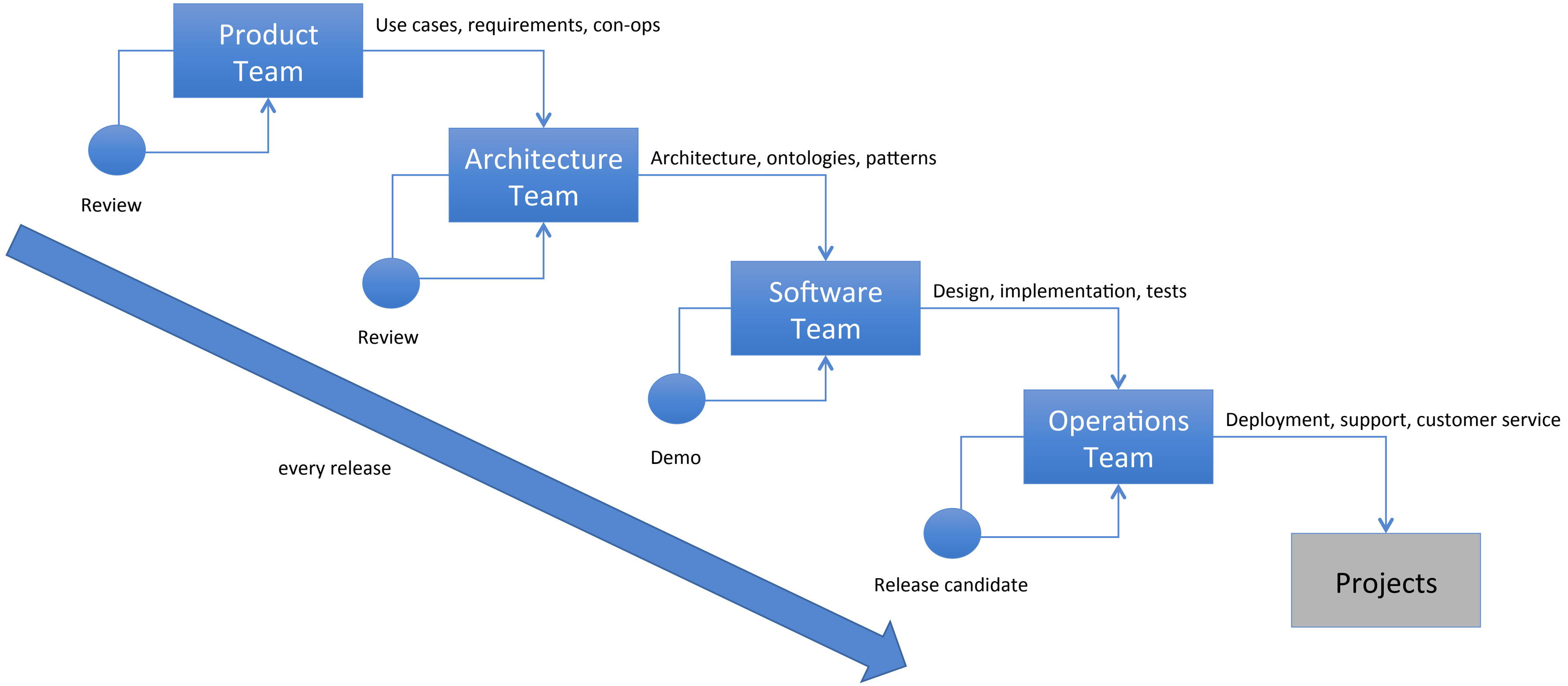
Reply [SJ]: what about…?

Software architecture



Architecture realization

Development organization and process



Product Team

Use cases, requirements, con-ops

Review

Architecture Team

Architecture, ontologies, patterns

Review

Software Team

Design, implementation, tests

Demo

every release

Operations Team

Deployment, support, customer service

Release candidate

Projects

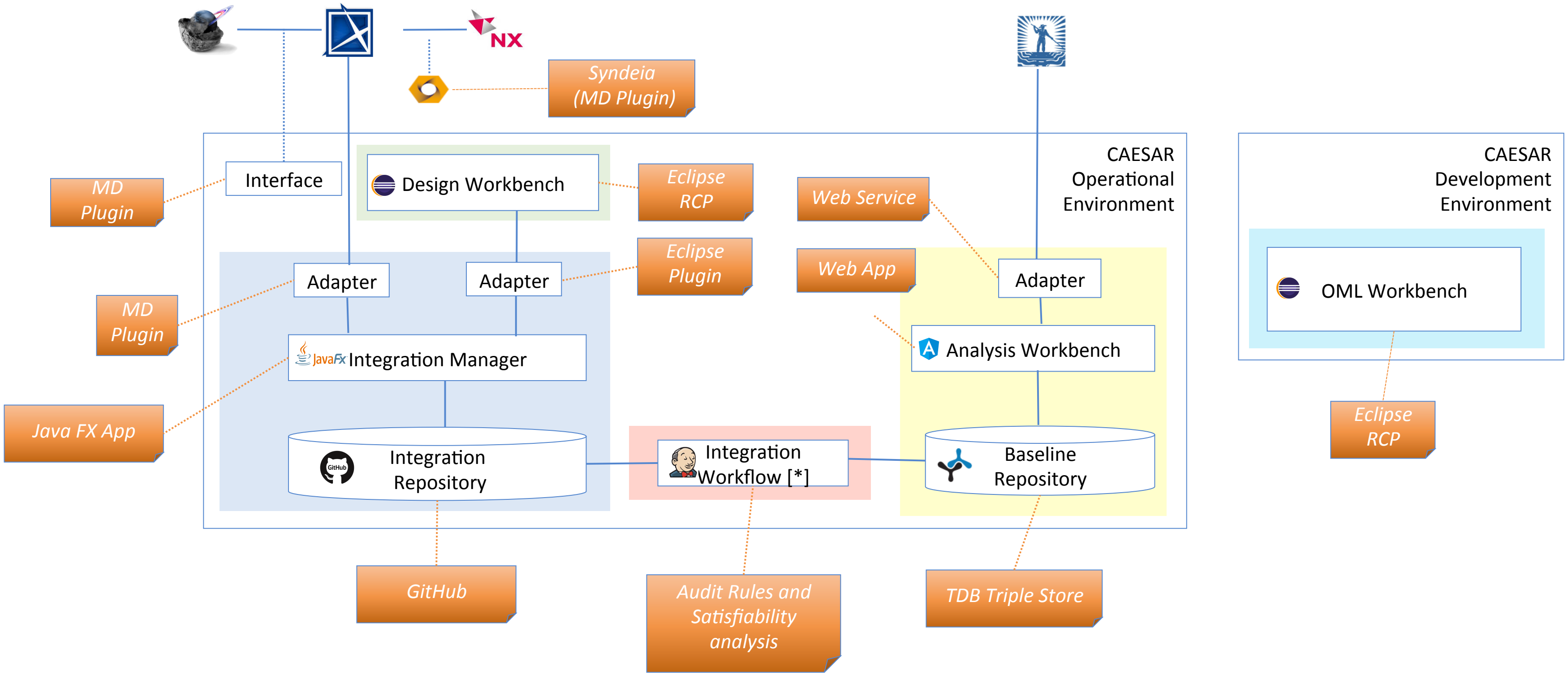
Development principles

- Incremental need-based scope
  - Develop requirements and capabilities incrementally over time
  - Develop use cases around discipline functions based on perceived need vs development risk/cost
  - Have a big list of discipline use cases to draw from, and haven't identified them all yet
- Resource-constrained effort
  - The process is resource-driven based on expected available budget and people
    - available resources determine how fast we can work off use cases and deliver products
- Agile development process
  - Short iterations or sprints to develop discrete features
  - Measure progress and get quick/frequent feedback from users
  - Helps predict future development performance
- Standard versioning scheme (major.minor.service.qualifier)
  - The major segment indicates breakage in the API
  - The minor segment indicates "externally visible" changes
  - The service segment indicates bug fixes
  - The qualifier segment indicates a particular build

Prototype (demo 1)

- CAESAR software team prototyped many use cases of the new architecture using a case study of **mass management**
- Demonstrated Workflows:
  1. Follow a methodology to author the system model using multiple tools
     - Start by finding a point design (in IME), elaborate the design (in MD), refine the design by importing an implementation (in NX)
  2. Incrementally change the system model baseline
     - Propose changes to the system model baseline (in integration repository)
     - Check system model consistency and correctness (with reasoner/audit)
     - Review change deltas in system model for suitability/relevance
     - Accept the changes and publish the revised model baseline for analysis
  3. Report on the system model baseline at any point
     - Publish a (Tom Sawyer) perspective to visualize the design details in the baseline
  4. Propose a change to the system model baseline based on an ECR
     - Perform changes to the system model baseline in another tool
     - Propose the changes back to the main tool and apply it there

Implementation architecture (prototype)

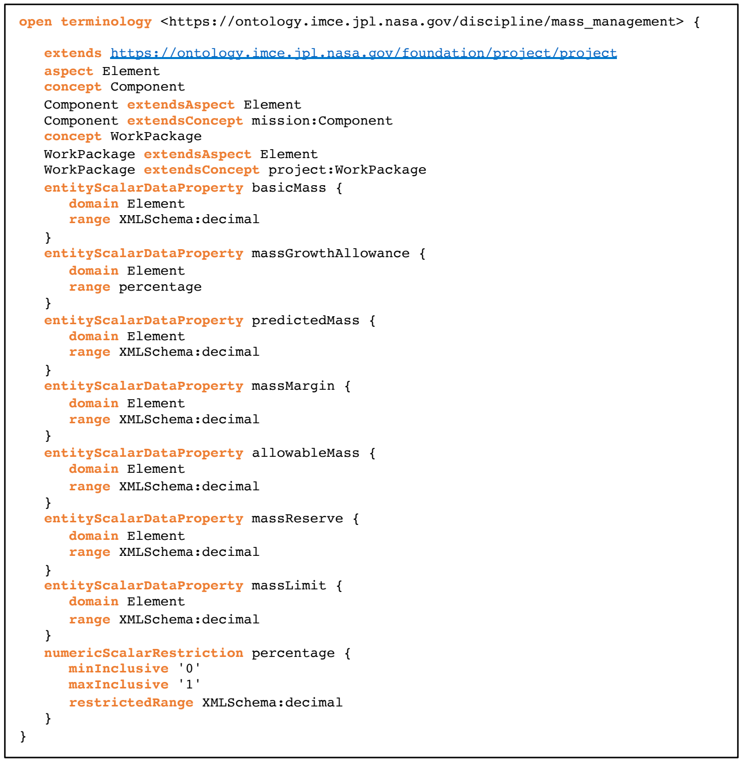


OML

- OML is implemented as a EMF-based domain specific language:
  - Abstract syntax (based on a subset of OWL) defined with an Ecore model
  - Textual syntax (more concise than RDF) defined with Xtext grammar
  - Generated and reflective Java API corresponding to abstract syntax
  - Ability to leverage the EMF tool ecosystem (query, view, transformations, persistence, compare/merge, etc.)
- OML can be translated to OWL DL allowing it to leverage semantics web technologies
  - Storage in triple stores (TDB)
  - Reasoning with over the shelf reasoners (Pellet)
  - Query with SPARQL
- OML can be translated into tabular format allowing scalable analysis
  - Efficient representation (in JSON)
  - Functional API (in Scala)
  - Distributed processing (with Spark)

Example: Mass management ontology

- Capture the system physical hierarchy using components and allocate them to work packages.
- Specify mass budgets on both component and work package level

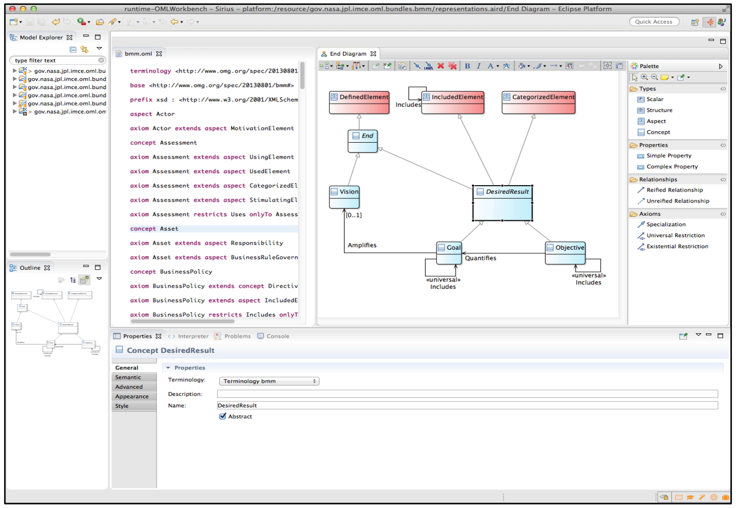


```
open terminology <https://ontology.imce.jpl.nasa.gov/discipline/mass_management> {

   extends https://ontology.imce.jpl.nasa.gov/foundation/project/project
   aspect Element
   concept Component
   Component extendsAspect Element
   Component extendsConcept mission:Component
   concept WorkPackage
   WorkPackage extendsAspect Element
   WorkPackage extendsConcept project:WorkPackage
   entityScalarDataProperty basicMass {
      domain Element
      range XMLSchema:decimal
   }
   entityScalarDataProperty massGrowthAllowance {
      domain Element
      range percentage
   }
   entityScalarDataProperty predictedMass {
      domain Element
      range XMLSchema:decimal
   }
   entityScalarDataProperty massMargin {
      domain Element
      range XMLSchema:decimal
   }
   entityScalarDataProperty allowableMass {
      domain Element
      range XMLSchema:decimal
   }
   entityScalarDataProperty massReserve {
      domain Element
      range XMLSchema:decimal
   }
   entityScalarDataProperty massLimit {
      domain Element
      range XMLSchema:decimal
   }
   numericScalarRestriction percentage {
      minInclusive '0'
      maxInclusive '1'
      restrictedRange XMLSchema:decimal
   }
}
```

Mass Management Ontology in OML

OML workbench

- Prototype
  - Provides text editor to author OML ontologies
  - Allows visualization of OML ontologies with graphical notation
  - Allows import/export of OML models from/to OWL DL
- Next steps
  - Provide OML documentation and tutorial
  - Provide various OML examples to showcase capabilities
  - Allow creation of OML ontologies with graphical notation
  - Provide a mapping framework to tool-specific representations
  - Provide a UI to reason on ontologies with Pellet



runtime–OMLWorkbench – Sirius – platform:/resource/gov.nasa.jpl.imce.oml.bundles.bmm/representations.aird/End Diagram – Eclipse Platform

```
terminology <http://www.omg.org/spec/20130801
base <http://www.omg.org/spec/20130801/bmm#>
prefix xsd : <http://www.w3.org/2001/XMLSchem
aspect Actor
axiom Actor extends aspect MotivationElement
concept Assessment
axiom Assessment extends aspect UsingElement
axiom Assessment extends aspect UsedElement
axiom Assessment extends aspect CategorizedEl
axiom Assessment extends aspect StimulatingEl
axiom Assessment restricts Uses onlyTo Assess
concept Asset
axiom Asset extends aspect Responsibility
axiom Asset extends aspect BusinessRuleGovern
concept BusinessPolicy
axiom BusinessPolicy extends concept Directiv
axiom BusinessPolicy extends aspect IncludedE
axiom BusinessPolicy restricts Includes onlyT
```

Concept DesiredResult

Terminology: Terminology bmm

Description:

Name: DesiredResult

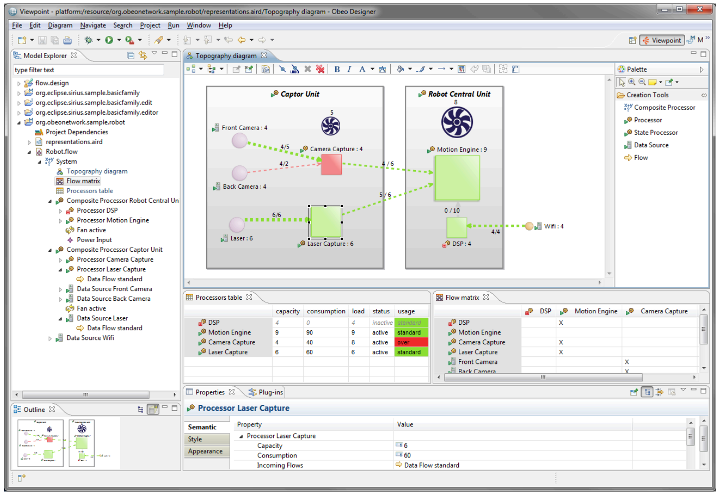
☑ Abstract

Tool Adapters

- Prototype
  - Provides profile generator as command-line tools invoked by a CI job
  - Bundles IMCE profile (that represents the IMCE terminologies) in an MD plugin
  - Provides an exporter of SysML models (with IMCE profile) to OML models
- Next steps
  - Provide profile generator (from OML terminologies) as a MD plugin
  - Provide MD client adapter as MD plugin
    - Provide IMCE profile as either a deployed file or in TWS/TWC
    - Provide validation support for IMCE profiles within MD
    - Provide an exporter for OML files
    - Provide an importer for OML files (with compare/merge support)
  - Provide MD server adapter as a web service
    - Provide an OML exporter as a web service
  - Provide adapters to other tools (e.g., Capital)

Design workbench

- Prototype
  - Implements design workbench as an Eclipse RCP
  - Provide native vocabulary representation for as Ecore model
  - Provides mass management perspective including mass management editor
  - Provides OML importer into native format
  - Persists OML files in github
- Next steps
  - Provide client adapter as Eclipse plugin
    - Provide importer/exporter of OML files
    - Provide validation support for native format
  - Provide server adapter as a web service
    - Provide an OML exporter as a web service
  - Provide discipline specific perspectives (e.g., EFSE)
    - Investigate interfacing with Xcel



Processors table

	capacity	consumption	load	status	usage
DSP	4	0	4	inactive	standard
Motion Engine	9	90	9	active	standard
Camera Capture	4	40	8	active	over
Laser Capture	6	60	6	active	standard

Flow matrix

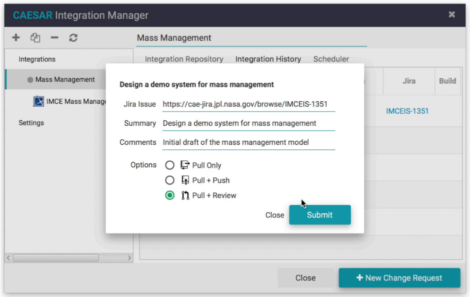
	DSP	Motion Engine	Camera Capture
DSP		X	
Motion Engine			
Camera Capture		X	
Laser Capture		X	
Front Camera			X
Back Camera			X

Processor Laser Capture

Property	Value
Processor Laser Capture	
Capacity	6
Consumption	60
Incoming Flows	Data Flow standard

Integration manager

- Prototype
  - Implements IM as a JavaFX application with settings stored in MySQL database
  - Supports creating/launching integration configurations
  - Supports integrating with Jira for issue management, with github for baseline management and with Jenkins for build management
  - Supports textual diff of OML files
- Next steps
  - Develop IM as a web app (backend: Play, frontend: React, database: MySQL)
  - Develop better UI to abstract out Jira, github and jenkins
  - Orchestrate integrations as workflows (assess Apache Camel)
  - Support multi-tenancy and role based access to the app
  - Support the ECR process better (distinguish it form integration)
  - Support the compare/merge process better (e.g., structured diff)
  - Define file organization and branching strategy for baseline in github



Design a demo system for mass management

Jira Issue: https://cae-jira.jpl.nasa.gov/browse/IMCEIS-1351

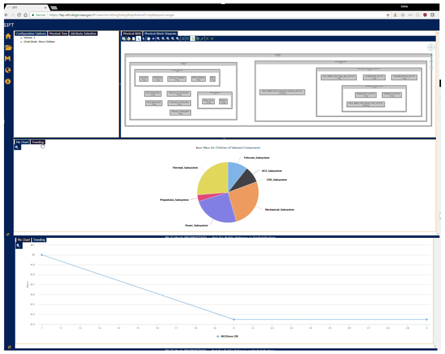
Summary: Design a demo system for mass management

Comments: Initial draft of the mass management model

Options: ○ Pull Only ○ Pull + Push ● Pull + Review

Analysis workbench

- Prototype
  - Implements a baseline repository as a TDB triple store
  - Reuses the SIFT web app (implemented in Angular) as the AWB
  - Provides a mass management perspective defined in Tom Sawyer
- Next steps
  - Develop a highly scalable baseline repository with ontology-based query API (e.g., Spark database)
  - Adopt a modern web architecture (front-end: React, backend: Play, database: MySQL)
  - Allow scalable text search (investigating elastic search)
  - Allow multi-tenancy and role-based access to repository
  - Support a the definition and registration of analysis workflows
  - Support a customizable dashboard reporting capability
  - Provide interfaces to relevant analysis/reporting OTS tools



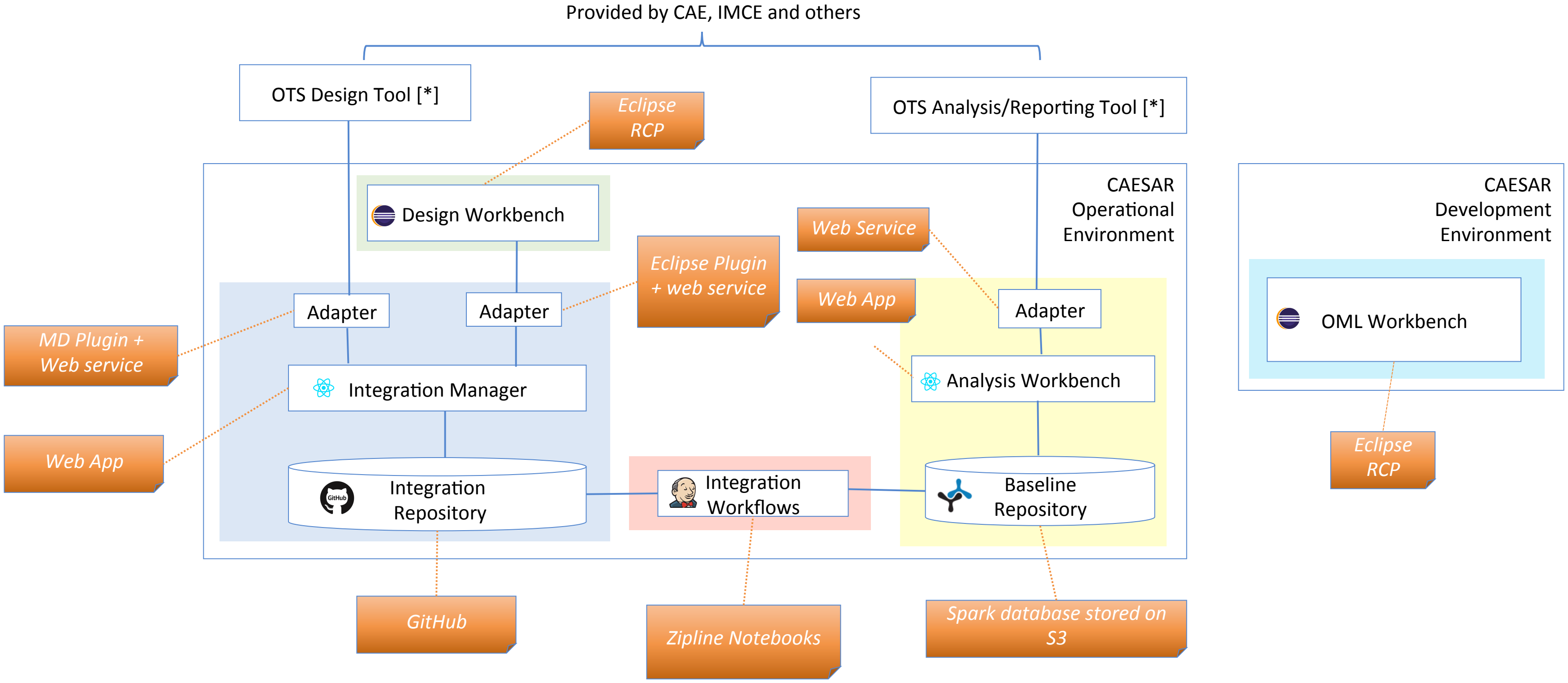
Integration/analysis workflows

- Prototype
  - Provides an OML to OWL converter as a command-line tool
  - Implements satisfiability analysis with Pellet as command-line tool
  - Implements audit rules with SPARQL as command-line tool
  - Launches integration workflows via Jenkins CI jobs
  - Deploys integration workflows in Docker container
- Next steps
  - Publish the baseline revisions to the new baseline repository (e.g., Spark database)
  - Re-implement the audit rules in a scalable way with the repository's new API (e.g., SQL)
  - Provide better reporting mechanisms for errors
  - Investigate notebooks (e.g., Zipline) as a technology for integration workflows
  - Allow registration and customization of integration workflows per project

Prototype lessons learned

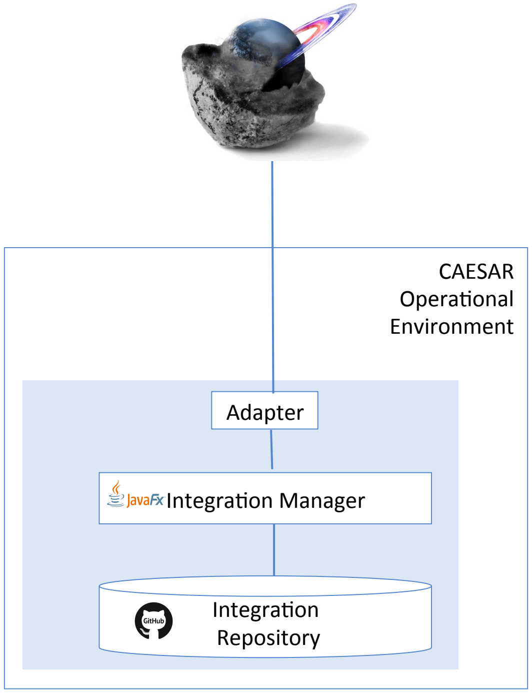
- Architecture is realizable; some technical challenges were identified and will be looked into further; some new components were discovered (e.g., integration manager)
- Coming to common understanding on discipline-specific processes and ontologies take time; need to work closely with SMEs
- OML is promising as a tool-neutral representation of information and to support model integration/analysis; need more docs, examples, trainings
- Foundational ontologies are good base to build discipline ontologies on; need more example and more regression tests
- The tool Syndeia works well when concepts are well aligned between tools; otherwise customization is needed (but not always possible); explore building OML based linking tool
- EMF technology stack made developing OML WB an Design WB easier
- Interfacing with Foundry IMCE is doable; need to schedule completing it
- Interfacing with COTS can be tricky; need to work closely with and influence vendors

Implementation architecture



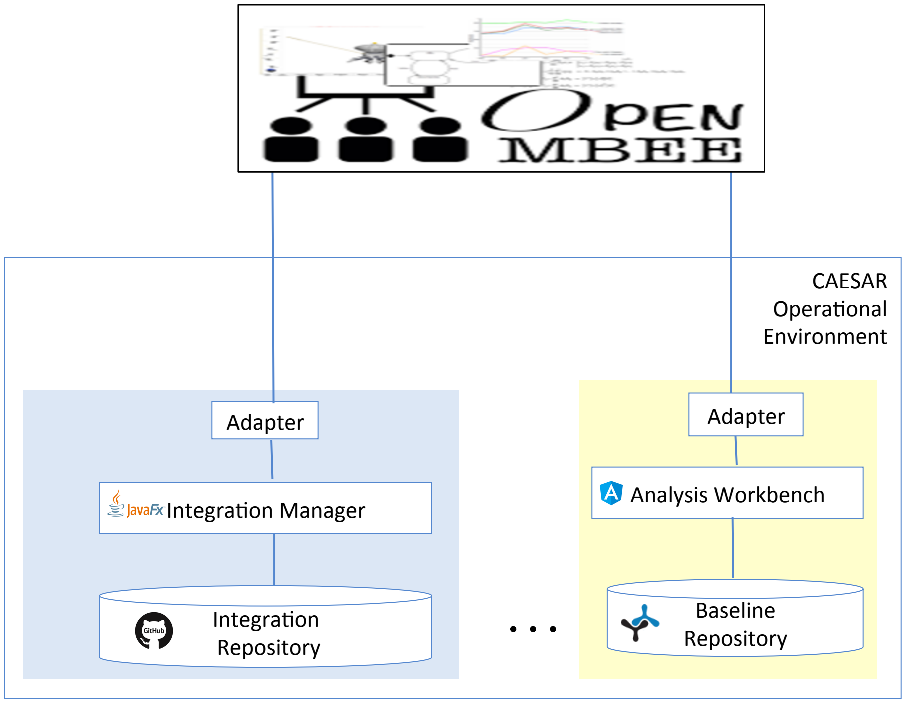
Integration with Foundry IME

- Foundry IME is a system modeling and analysis environment for the formulation phase of a project
- CAESAR can integrate with the Foundry IME by developing a tool adapter that translates models in IME to OML



Integration with CAE OpenMBEE

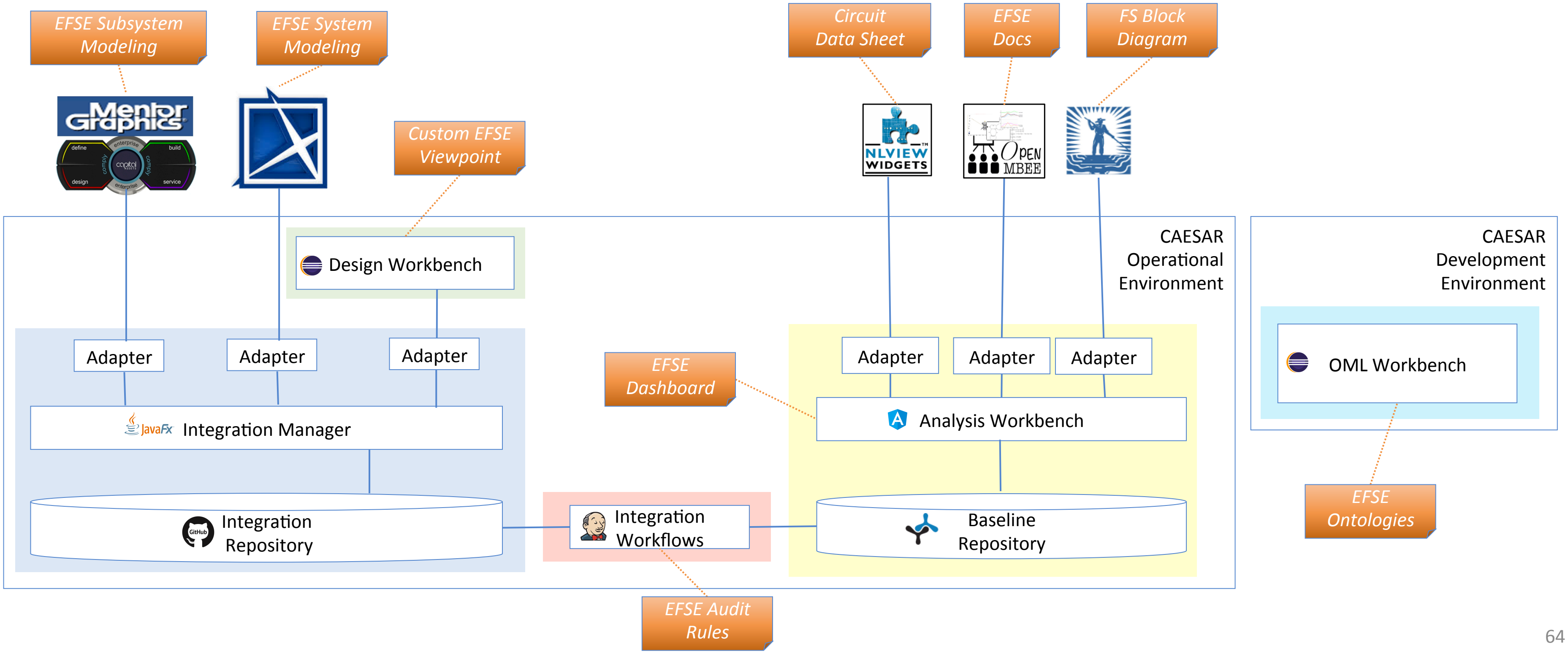
- CAE OpenMBEE is an application for producing document viewpoints from system models
- CAESAR can integrate with CAE OpenMBEE as a design tool by implementing an adapter that translates its models in MMS to OML
- CAESAR can integrate with CAE OpenMBEE as a reporting tool by implementing an adapter that translates OML models to MSS



CAESAR 2.0 (initial FY18) release scope

- CAESAR 2.0 release will focus on addressing some EFSE use cases
  - Milestone 1
    - Simplify authoring of EFSE system model by developing a custom UI (e.g., tabular viewpoint)
    - Develop an interface between the custom UI and MagicDraw
  - Milestone 2
    - Develop an interface from MagicDraw to the Capital tool suite (subsystem modeling tools)
    - Develop a methodology for using the different tools in the Capital suite
  - Milestone 3
    - Develop an interface from Capital Logic to NL View to visualize circuit data sheets
    - Develop equivalent CAESAR 1.x gate products for EFSE
- CAESAR 2.x releases may add other use cases
  - e.g., behavior modeling, integration to FP/FCR tool

CAESAR 2.0 implementation architecture



Open source strategy

- CAESAR components will be closed source (hosted on JPL github) by default
  - This protects JPL investment and addresses IP and ITAR concerns
- However, some components will be open sourced (hosted on public github)
  - Those will mostly be infrastructure type components
  - Those will enable IMCE to build partnerships with industry
  - Those ones will enable leveraging open source / academic ecosystems
  - Those include so far: OML tools, ontologies, and some tool adapters
- Some components may directly be developed by 3rd party vendors
  - Ideally, we want to delegate vendor-specific work to the vendors

Operations architecture

- A detailed operation plan for CAESAR 2 is still **work in progress**
- Packaging
  - Package enterprise applications in versioned Docker containers (for easy deployment)
  - Package desktop applications (including COTS provided by CAE) with their plugins in versioned bundles
    - Enumerate supported OSs and environments for each application
    - Automatic detection of new versions with prompt to install them
  - Provide library models in versioned files or configuration-controlled model repositories
- Deployment
  - Provide lab-wide deployments for major (breaking) releases (e.g., v2, v3) of enterprise applications
    - Perform automatic migration for minor (new features) and patch releases (bug fixes)
    - Support major releases until all projects migrate to next major release
    - Separate deployments of major releases may be setup on the foreign national network (for non ITAR sensitive projects)
  - Projects can request their own private deployments (at additional cost)
  - Enterprise applications will run on OCIO-supported Gov Cloud virtual machines
  - Desktop applications will run on project members' (physical or virtualized) workstations
- Configuration
  - Manage deployment wide settings (and accounts) by IMCE team
  - Manage project-specific settings (and accounts) by projects
    - IMCE consultants may be hired by projects to manage these settings

Operations architecture (cont'd)

- Customization
  - IMCE team will deliver incremental discipline features in major and minor releases
  - IMCE consultants may be hired by flight projects to develop project-specific features
  - CAESAR will publish API to allow projects to develop extensions themselves if desired
- Support
  - IMCE team will provide a Jira project to receive support issues from projects
    - Questions (on using, deploying, configuring CAESAR)
    - Bugs (reporting problems found in CAESAR)
    - Stories (feature or task requests)
- Training
  - IMCE will develop training modules on the various features of CAESAR
  - IMCE will have regular scheduled trainings and will provide others on demand

CAESAR flight project sandbox

- CAESAR operation team plans to create a flight project sandbox to test capabilities in before deployment
- The sandbox will exercise end-to-end modeling and analysis use cases

Project infusion plan

- CAESAR mainly targets large to medium sized projects at JPL
  - CAESAR product team will work closely with projects to prioritize requirements
  - Incremental capabilities/releases will align with project milestones
    - E.g., CAESAR 2.0 release targets ESE capabilities for Europa
  - IMCE will provide reusable model libraries to jump start new project modeling effort
    - E.g., project template MD model, MD library model based on ontologies, cookbook model
  - IMCE will work with line to train their personnel on CAESAR
- CAESAR will support smaller projects at JPL as a secondary target
  - This will be achieved by streamlining (or scaling down) the processes when necessary
- CAESAR may later support projects at other organizations
  - This would allow fostering relationships with strategic partners/sponsors

Project infusion plan

- CAESAR mostly provides end-user products to be used by systems engineers
  - I.e., there will not be expectation of a project to have software team to use CAESAR
- CAESAR will also provide extension points that will allow software savvy developers to augment the feature set of CAESAR
  - CAESAR will publish and document a set of API for doing that
  - This can be used by projects with software savvy teams (or by vendors) to add features to CAESAR